使用者指南

AWS 大型主機現代化

AWS 大型主機現代化: 使用者指南

Table of Contents

什麼是 AWS 大型主機現代化 1
AWS 大型主機現代化的特色 2
模式 2
如何開始使用 AWS 大型主機現代化 3
相關服務 3
存取 AWS 大型主機現代化 4
您是第一次使用 AWS 大型主機現代化的使用者嗎？ 4
AWS 大型主機現代化的定價 4
設定 AWS 大型主機現代化 5
註冊一個 AWS 帳戶 5
建立具有管理存取權的使用者 5
概念 7
應用程式 7
應用定義 7
Batch 工作 8
組態 8
資料集 9
環境 9
大型主機現代化 9
移民旅程 9
掛載點 9
自動重構 9
平台轉換 9
資源 9
运行时引擎 10
現代化方法 11
評估階段 11
動員階段 11
遷移和現代化階段 12
操作和優化相位 12
開始使用 13
教程：為 AWS 藍光時代設置託管運行時 13
必要條件 13
步驟 1：上傳演示應用程序 14

步驟 2：建立應用程式定義 14
步驟 3：建立執行階段環境 15
步驟 4：建立應用程式 19
步驟 5：部署應用程式 21
步驟 6：啟動應用程式 23
步驟 7：訪問應用程序 24
步驟 8：測試應用程式 25
清除資源 26
教學課程：設定 Micro Focus 的受管理執行階段 26
必要條件 27
步驟 1：建立並載入 Amazon S3 儲存貯體 28
步驟 2：建立和設定資料庫 29
步驟 3：建立並設定 AWS KMS key 31
步驟 4：建立和設定 AWS Secrets Manager 資料庫密碼 32
步驟 5：建立執行階段環境 33
步驟 6：建立應用程式 39
步驟 7：部署應用程式 45
步驟 8：匯入資料集 47
步驟 9：啟動應用程式 53
步驟 10：Connect 到 CardDemo CICS應用程序 54
清除資源 61
後續步驟 62
AWS 藍色時代重構 63
AWS 藍光時代版本資訊 64
發行版本公告 4.2.0 65
執行階段版本 4.2.0 66
現代化工具 4.2.0 版 70
發行版本公告 4.1.0 71
執行階段版本 4.1.0 72
現代化工具 4.1.0 版 76
發行版本公告 4.0.0 77
執行階段版本 4.0.0 78
現代化工具 4.0.0 版 83
發行版本公告 86
執行階段版本 3.10.0 86
現代化工具版本 3.10.0 88

發行版本公告 90
執行階段版本 3.9.0 90
現代化工具 3.9.0 版 95
發行版本公告 97
執行階段版本 3.8.0 98
現代化工具 3.8.0 版 100
發行版本公告 103
執行階段版本 3.7.0 103
現代化工具 3.7.0 版 105
發行版本公告 3.6.0 107
執行階段版本 3.6.0 108
現代化工具 3.6.0 版 111
發行版本公告 3.5.0 113
執行階段版本 3.5.0 114
現代化工具 3.5.0 版 116
升級 AWS 藍光時代 118
從 3 月 10 日移轉至 4.0.0 版 119
AWS 藍色時代運行時概念 120
高階架構 120
現代化應用程式結構 124
瞭解資料簡化器 158
AWS 藍色時代布魯桑 165
藍色管理主控台 182
AWS 藍色時代運行時配置 215
應用程式組態基 216
應用優先權 218
JNDI用於資料庫 218
AWS 藍光時代運行時秘密 218
其他文件（常規，SQL 等） 229
其他 Web 應用程式 230
啟用屬性 230
可用的 Redis 的緩存屬性 273
設定 Gapwalk 應用程式的安全性 286
AWS 藍色時代運行時 APIs 302
用於建置的端點 URLs 302
間隙行走應用程式的端點 303

Blusam 應用程式主控台端點 REST 320
管理JICS應用主控台 343
数据结构 361
設置 AWS 藍光時代運行時（非託管） 370
AWS 藍色年齡運行時先決 371
入職 AWS 藍光時代運行時 371
基礎架構設定需 376
在 Amazon 上部署 AWS 藍光時代運行 ECS 382
在 Amazon 上部署 AWS 藍光時代運行 EC2 390
測試 PlanetsDemo 應用程式 403
使用藍光時代開發人員修改源代碼 IDE 406
教程：為 AWS 藍光時代開發人員設置 AppStream 2.0 IDE 407
教程：在 AppStream 2.0 上使用 AWS 藍光時代開發人員 411
微焦點重塑 429
設置微型聚焦運行時間（在 Amazon 上EC2） 429
微焦點運行時（在 Amazon 上EC2）先決條件 430
為 Amazon S3 創建亞馬遜VPC端點 430
要求帳戶的允許清單更新 432
建立角 AWS Identity and Access Management 色 433
授予 License Manager 所需的權限 440
訂閱 Amazon 機器映像 441
啟動微焦點實體 444
子網路或沒VPC有網際網路存取 449
設定 AppStream 2.0 自動化 456
在工作階段開始時設定自動化 456
在工作階段結束時設定自動化 457
在企業開發人員中檢視資料集做為表格 457
必要條件 458
步驟 1：設定與 Micro Focus 資料存放區的ODBC連線 (Amazon 資RDS料庫) 458
步驟 2：建立 MFDBFH .cfg 檔案 460
步驟 3：為您的字帖版面建立結構 (STR) 檔案 461
第 4 步：使用結構（STR）文件創建數據庫視圖 463
步驟 5：以表格和欄的形式檢視微型焦點資料集 464
微焦點教程 464
教學課程：設定 BankDemo範例應用程式的組建 465
教學課程：與微焦點企業開發者建立 CI/CD 管道 474

教學課程：為企業分析器和企業開發人員設定 AppStream 2.0 497
教學課程：搭配企業開發人員使用 506
教學課程：設定企業分析器 516
教學課程：設定企業開發人員 526
Batch 公用程 531
二進位位置 531
M2 SFTP 批次工具 532
M2 WAIT 批次工具 538
TXT2PDF批次程式 540
M2 DFUTIL 批次工具 545
M2 RUNCMD 批次工具 552
精確複製資料 556
必要條件 556
訂閱 Amazon 機器映像 556
使用 AWS 精確地啟動大型主機現代化資料複製 557
建立IAM策略 558
建立 IAM 角色 558
將IAM角色附加到 Amazon 執EC2行個體 559
彙編程序轉換 mLogica 560
什麼是彙編程序轉換 mLogica 560
代碼轉換編譯器 561
程式碼轉換架構 561
自動化方法 562
安全 562
其他資源 562
瞭解程式碼轉換計費 562
代碼轉換計費和範圍 562
程式碼轉換概念 564
巨集處理 565
字碼頁 (EBCDICvsASCII) 565
CodeBuild 565
了解組件和過程 565
AWS Mainframe Modernization 容器 565
S3 專案儲存貯體 566
記錄檔位置 566
程序概觀 567

教學課程：將程式碼從彙編程式轉換為 COBOL 567
必要條件 568
步驟 1：共用組建資產 AWS 帳戶 568
步驟 2：創建 Amazon S3 存儲桶 569
步驟 3：建立IAM策略 569
步驟 4：建立IAM角色 571
步驟 5：將IAM策略附加到IAM角色 572
步驟 6：建立 CodeBuild 專案 572
第 7 步：定義項目並上傳源代碼 578
步驟 8：執行分析並瞭解報告 579
步驟 9：執行程式碼轉換 580
步驟 10：驗證代碼轉換 583
步驟 11：下載轉換後的代碼 584
清除資源 585
查龍集成 586
介紹查龍 SSP 586
支援的客體作業系統 587
Charon SSP 雲端執行個體先決條件 588
實例先決條 589
為 Charon 建立和設定 AWS 雲端執行個體 (新增GUI) 590
一般先決條件 590
使用啟 AWS Management Console 動新執行個體 591
使用重新平台 NTT DATA 596
必要條件 596
訂閱 Amazon 機器映像 596
使用執行個體啟動 AWS 大型主機現代化重新平台 NTT DATA 596
開始使用NTT資料 597
受管理應用 599
為移轉的應用程式建立 AWS 資源 600
所需的許可 600
Amazon S3 儲存貯體 600
資料庫 601
AWS Key Management Service 鍵 601
AWS Secrets Manager 秘密 602
建立應用程式 602
建立應用程式 602

部署應用程式 603
部署應用程式 603
更新應用程式 604
更新應用程式 604
刪除應用程式 605
刪除應用程式 605
提交應用程式的批次工作 606
提交批次工作 606
重新啟動批次工作 607
取消應用程式的批次工作 608
取消批次處理工作 608
匯入應用程式的資料集 608
匯入資料集 609
管理應用程式的交易 609
管理應用程式的交易 609
設定管理的應用程式 610
AWS 藍光時代管理應用程式的結構 611
設定受管理應用程式之公用程式的 612
為受管理的應用程式設定其他 622
應用定義參考 642
。一般頭部分。 642
定義區段概觀 644
AWS 藍光時代的應用程序定義 644
AWS 藍光時代定義詳情 645
微焦點應用定義 650
微焦點定義細節 651
資料集定義參考 658
一般屬性 659
資料集請求格式範例 VSAM 660
GDG基礎的範例資料集請求格式 662
PS 或GDG層代的範例資料集請求格式 663
PO 的資料集請求格式範例 664
管理運行時環境 666
建立執行階段環境 666
建立執行階段環境 666
更新執行階段環境 669

更新執行階段環境 669
Maintenance window (維護時段) 669
停止執行階段環境 671
停止執行階段環境 671
重新啟動執行環境 672
重新啟動執行環境 672
刪除執行階段環境 672
刪除執行階段環境 672
應用程式測試 674
什麼是應用程序測試 674
您是第一次使用應用程式測試嗎？ 675
應用程序測試的好處 675
與整合 AWS CloudFormation 675
應用程式測試的運作 676
相關服務 3
訪問應用測試 677
應用程式測試的定價 677
應用程式測試概 678
測試用例 678
測試套件 679
測試環境配置 679
上傳 679
重新播放 679
Compare 680
資料庫比較 680
資料集比較 680
比較狀態 680
等價規則 681
最終狀態資料集比較 681
狀態進度資料庫比較 681
功能等價 (FE) 681
在線屏幕比較 682
重播資料 682
參考資料 682
上傳、重播和比較 682
差異 683

等值 683
來源應用 683
目標應用 683
應用程序測試先 683
應用程式測試中的主控台 684
在應用測試中創建測試用例 684
在應用測試中創建測試套件 686
在應用程序測試中創建測試環境配 688
教學：在 CardDemo 應用程式測試中設定應用程式 690
必要條件 690
步驟 1：準備設定 CardDemo 690
步驟 2：建立所有必要的資源 691
步驟 3：部署並啟動應用程式 691
步驟 4：匯入初始資料 692
步驟 5：Connect 到 CardDemo應用程序 693
教程：使用 AWS 藍光時代重播和比較 CardDemo 694
步驟 1：獲取 AWS 藍光時代 Amazon EC2 Amazon 機器圖像（AMI） 694
步驟 2：使用 AWS 藍光時代啟動 Amazon EC2 實例 AMI 694
步驟 3：將 CardDemo 相依檔案上傳至 S3 695
步驟 4：載入資料庫並初始化 CardDemo應用程式 696
步驟 5：啟動 AWS 藍光時代運行時 CloudFormation 698
步驟 6：測試 AWS 藍光時代 Amazon EC2 實例 700
步驟 7：驗證之前的步驟是否正確完成 701
步驟 8：創建測試用例 702
步驟 9：建立測試套件 702
步驟 10：建立測試環境設定 703
步驟 11：在測試套件中上傳輸入數據 703
步驟 12：重播和比較 704
應用程式測試中支援的資料集字碼頁 704
在應用程序測試數據保護 713
AWS 大型主機現代化應用程式測試所收集的資料 714
AWS 大型主機現代化應用程式測試的靜態資料加密 715
建立客戶受管金鑰 715
為 AWS 大型主機現代化應用程式測試指定客戶管理的金鑰 716
AWS 大型主機現代化應用程式測試加密內容 717
監控加密金鑰 718

傳輸中加密 718
檔案傳輸 719
什麼是文件傳輸 719
AWS大型主機現代化檔案傳輸的優點 719
AWS大型主機現代化檔案傳輸的運作方式 720
安裝檔案傳輸代理程式 721
步驟 1：為 M2 代理程式建立 ZF 資料集 721
步驟 2：將資料集格式化為 ZF 721
步驟 3：掛載檔案系統 722
步驟 4：驗證掛載 722
步驟 5：輸入 OMVS 722
步驟 6：設定代理程式安裝目錄環境變數 722
步驟 7：設定工作目錄環境變數 722
步驟 8：建立工作目錄 723
步驟 9：複製代理程式 tar 檔案並複製工作目錄 723
步驟 10：假設根使用者 723
步驟 11：完成代理程式安裝 723
設定檔案傳輸代理程式 724
步驟 1：配置權限和啟動任務控制（STC） 724
步驟 2：創建 Amazon S3 存儲桶 725
步驟 3：建立用於加密的 AWS KMS 客戶管理金鑰 725
步驟 4：建立大型主機認證的 AWS Secrets Manager 密碼 726
步驟 5：建立IAM策略 727
步驟 6：建立具有長期存取認證的IAM使用者 728
步驟 7：建立代理程式要承擔的IAM角色 729
步驟 8：代理程式組態 730
建立資料傳輸端點 732
建立資料傳輸端點 732
建立移轉任務 734
建立移轉任務 734
檢視移轉任務 737
教學課程：開始使用檔案傳輸 737
概觀 737
步驟 1：將代理程式二進位檔 tar 套件從傳輸 AWS 到大型主機邏輯分割區 738
步驟 2：在來源大型主機上設定檔案傳輸代理程式 738
步驟 3：建立資料傳輸端點 738

步驟 4：建立轉移任務 738
步驟 5：查看轉移任務進度 738
支持的源代碼和目標代碼頁 738
大型主機資料集類型 738
支持的代碼頁 738
安全 740
資料保護 741
AWS 大型主機現代化收集的資料 741
AWS 大型主機現代化服務的靜態資料加密 742
AWS 大型主機現代化如何使用補助金 AWS KMS 744
建立客戶受管金鑰 745
指定 AWS 大型主機現代化的客戶管理金鑰 747
AWS 大型主機現代化加密內容 748
監控加密金鑰 749
進一步了解 764
傳輸中加密 764
身分和存取權管理 764
物件 765
使用身分驗證 765
使用政策管理存取權 768
AWS 大型主機現代化如何搭配運作 IAM 770
身分型政策範例 782
故障診斷 784
使用服務連結角色 785
法規遵循驗證 788
恢復能力 789
基礎架構安全 789
AWS PrivateLink 790
考量事項 790
建立介面端點 790
建立端點政策 791
監控 793
使用監控 CloudWatch 793
執行環境測量結 793
應用程式指標 794
維度 798

記錄API呼叫 CloudTrail 798
AWS 大型主機現代化資訊 CloudTrail 799
瞭解 AWS 大型主機現代化記錄檔項目 800
M2 的疑難排解 802
疑難排解錯誤：等待資料集名稱解除鎖定時逾時 802
常見原因 802
解析度 803
強制鎖定釋放 803
配置布魯桑自 auto 修復機制 804
布魯薩姆鎖管理器 804
疑難排解錯誤：無法存取應用程式 URL 805
常見原因 805
解析度 805
疑難排解：無法從主控台開啟 AWS 藍光深入解析 806
常見原因 806
解析度 806
疑難排解錯誤：環境狀況不良 807
常見原因 807
解析度 807
疑難排解微焦點的授權問題 808
確認 Amazon EC2 執行個體具有IAM授權角色 808
使用可達性分析儀 809
執行授權守護程式 809
作業系統修補之後，Linux 上的企業伺服器或企業組建工具的授權問題 810
文件歷史紀錄 811
..... dcccxiii

什麼是 AWS 大型主機現代化？

AWS 大型主機現代化可協助您將大型主機應用程式現代化為受管理的執行階段環境。 AWS 它提供工具和資源來協助您規劃和實作遷移和現代化。您可以分析現有的大型主機應用程式、使用或 PL/I 進行開發COBOL或更新，以及實作自動化管道，以持續整合和持續交付應用程式 (CI/CD)。您可以根據客戶的需求，在自動重構和重新平台模式之間進行選擇。如果您是協助客戶移轉大型主機工作負載的顧問，則可以在移轉和現代化旅程的所有階段 (從初始規劃到移轉後 AWS 雲端作業) 使用大型主機現代化工具。

您可以使用 AWS 大型主機現代化，協助您有效率地為您的大型主機應用程式建立和管理執行階段環境，以及管理和監視現代化的應用程式。 AWS

主題

- AWS 大型主機現代化的特色
- 模式
- 如何開始使用 AWS 大型主機現代化
- 相關服務
- 存取 AWS 大型主機現代化
- 您是第一次使用 AWS 大型主機現代化的使用者嗎？
- AWS 大型主機現代化的定價

> **Note**
>
> 您是否曾與 AWS 大型主機移轉能力合作夥伴或 AWS 專業服務合作夥伴參與大型主機現代化專案？ 如果沒有，我們強烈建議您為您的項目聘請專家。
>
> - AWS 大型主機現代化能力合作夥伴
> - AWS Professional Services

大 AWS 型主機現代化的功能和使用案例支援進化現代化方法，透過改善敏捷性和大量後續最佳化和創新的機會，從而提供短期的勝利。如需詳細資訊，請參閱現代化方法。

AWS 大型主機現代化的特色

AWS 大型主機現代化功能支援下列使用案例：

- 評 AWS 估：大型主機現代化的評估功能可協助您評估、規劃移轉與現代化專案，以及規劃移轉與現代化專案。
- 重構：由 AWS Blu Age 提供支援，您可以使用重構來轉換舊版應用程式編程語言、建立巨型服務或微服務，以及現代化使用者介面 () UIs 和應用程式軟體堆疊。

  AWS 藍光見解現在可 AWS Management Console 透過單一登入取得。您不必再管理單獨的 AWS 藍光見解憑據。您可 AWS AWS 以直接從. AWS Management Console
- 重新平台：由 Micro Focus 企業解決方案提供支援，您可以將應用程式移植到大部分應用程式原始程式碼，而不需要變更。
- 開發人員IDE： AWS 大型主機現代化提供隨選整合式開發環境 (IDE)，讓開發人員可以透過智慧型編輯與偵錯、即時程式碼編譯和單元測試，更快速地撰寫程式碼。
- 受管理的執行階段： AWS 大型主機現代化受管理的執行階段環境會持續監控您的叢集，透過自我修復運算和自動化調整來保持企業工作負載
- 持續整合與交付 (CI/CD)： AWS 大型主機現代化的 CI/CD 功能可協助應用程式開發團隊更頻繁、更可靠地進行程式碼變更，進而加速移轉速度、提升品質，並有助於減少新的業務功能發佈。 time-to-market
- 與其他 AWS 服務整合：大 AWS 型主機現代化支援 AWS CloudFormation，以及可重複部署 AWS PrivateLink，以及更高 AWS Key Management Service 的安全性與合規性。
- 擴充可用性： AWS 大型主機現代化現已在美國東部 (俄亥俄)、美國西部 (加利佛尼亞北部)、亞太區域 (孟買)、亞太區域 (首爾)、亞太區域 (新加坡)、亞太區域 (東京)、歐洲 (倫敦) 和歐洲 (巴黎) 推出。

如需 AWS 大型主機現代化功能的詳細資訊，請參閱。https://aws.amazon.com/mainframe-modernization/features/

模式

由 AWS Blu Age 提供支援的自動化重構模式專注於透過將完整的舊版應用程式堆疊及其資料層轉換為現代 Java 應用程式，同時保留功能等效性，以加速現代化。在此自動化轉換期間，它會建立具有 Angular 前端、API啟用 Java 後端和存取現代資料存放區的資料層的多層應用程式。重構程序提供與舊版堆疊相同的功能，以提高專案自動化，進而提高速度、品質和更低的成本，以便更快地實現業務利益。如需詳細資訊，請參閱AWS 大型主機現代化自動重構。

由 Micro Focus Enterprise 套件提供支援的重新平台模式專注於保留應用程式語言、程式碼和成品，以便將對應用程式資產和團隊的影響降到最低。它可以幫助客戶保持應用知識和技能。雖然應用程序的更改是有限的，這種模式也有助於基礎設施和流程的現代化。基礎結構會變更為現代雲端託管服務，而程序會變更為遵循應用程式開發和 IT 作業的最佳實務。如需詳細資訊，請參閱AWS 大型主機現代化重新平台。

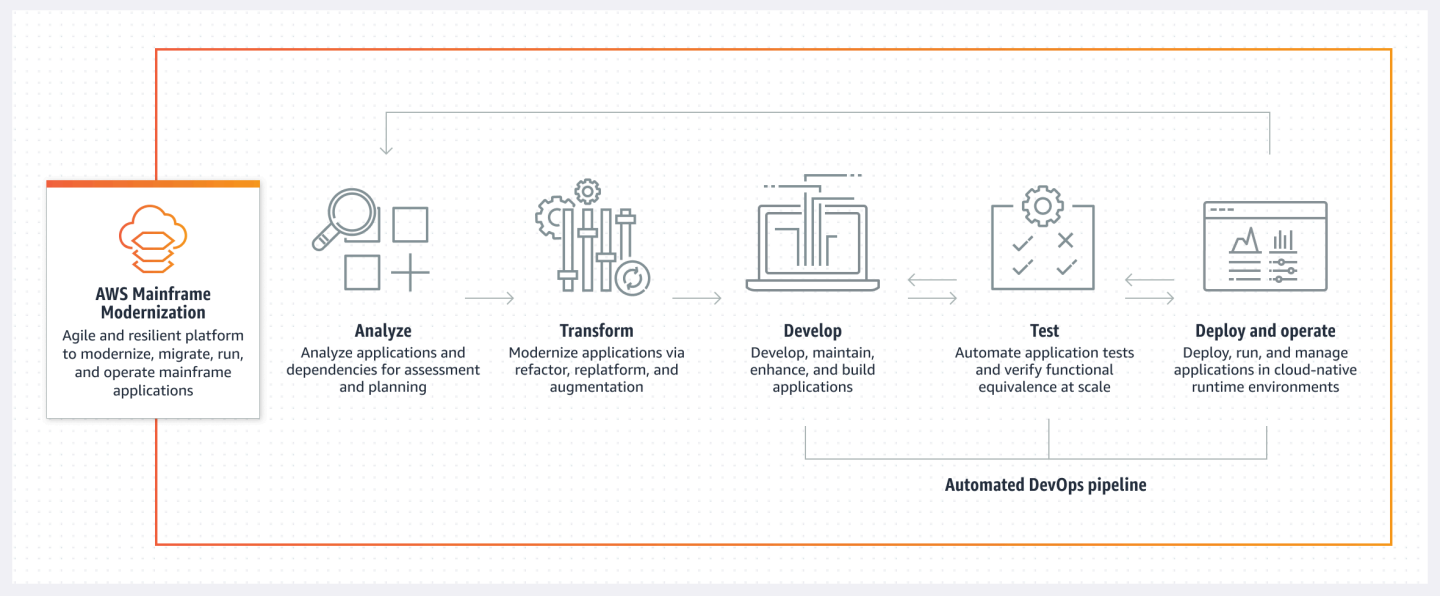
如何開始使用 AWS 大型主機現代化

試試看！ AWS 我們提供教學課程和範例應用程式，協助您瞭解大型主機現代化提供的功能。選擇教程：為 AWS 藍光時代設置託管運行時或以取得完整教學課程：設定 Micro Focus 的受管理執行階段的 step-by-step 自學課程。

如果您對自動重構感興趣，請查看 AWS Blu Age 工具，網址為。BluInsights您也可以設定 AppStream 2.0 以存取 AWS 藍光時代開發人員IDE，或 Micro Focus 企業分析器和 Micro Focus 企業開發人員工具。

教學課程和範例應用程式只能讓您瞭解 AWS 大型主機現代化所提供的功能。當您準備好開始現代化專案時，請參閱以現代化方法取得有關現代化專案階段和工作的詳細資訊。

下圖顯示 AWS 大型主機現代化服務用於分析、轉換、開發、測試和部署和操作大型主機應用程式的工作流程。



相關服務

除了用於自動化重構的 Blu Insights 之外，您還可以在大型主機現代化中使用以下 AWS 服務。 AWS

- Amazon 託RDS管您遷移的數據庫
- Amazon S3 用於存放應用程式二進位檔案和定義檔
- Amazon FSx 或 Amazon EFS 存儲應用程序數據
- Amazon AppStream 可以訪問 Micro Focus 企業分析儀和微焦點企業開發人員工具
- AWS CloudFormation 用於為移轉的應用 DevOps 程式設定 CI/CD 的自動化管道
- AWS Migration Hub
- AWS DMS 用於移轉您的資料庫

存取 AWS 大型主機現代化

目前，您可以透過主控台存取 AWS 大型主機現代化，位於。https://console.aws.amazon.com/m2/如需提供 AWS 大型主機現代化的區域清單，請參閱 AWS . Amazon Web Services 一般參考

您是第一次使用 AWS 大型主機現代化的使用者嗎？

如果您是 AWS 大型主機現代化的初次使用者，建議您先閱讀下列章節：

- 開始使用 AWS 大型主機現代化
- 設定 AWS 大型主機現代化

AWS 大型主機現代化的定價

AWS 大型主機現代化會根據支援受管執行階段環境的執行個體使用量收費。此外， AWS 大型主機現代化提供一些工具，無需額外付費。您需負責支付與 AWS 大型主機現代化相關的其他 AWS 服務所產生的費用。 AWS 將在使用 AWS 大型主機現代化的任何價格變更生效前 30 天通知。如需詳細資訊，請參閱使用. AWS

使用 AWS Blu 洞察，您需要支付轉型中心的使用費用。如需詳細資訊，請參閱AWS 大型主機現代化定價。

設定 AWS 大型主機現代化

在您開始使用 AWS 大型主機現代化之前，您或您的系統管理員必須註冊、使用系統管理設定建立使用者 AWS 帳戶，以及保護使用者的安全。IAM

主題

- 註冊一個 AWS 帳戶
- 建立具有管理存取權的使用者

註冊一個 AWS 帳戶

如果您沒有 AWS 帳戶，請完成以下步驟來建立一個。

若要註冊成為 AWS 帳戶

1. 打開https://portal.aws.amazon.com/billing/註冊。
2. 請遵循線上指示進行。

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   當您註冊一個時 AWS 帳戶，將創建AWS 帳戶根使用者一個。根使用者有權存取該帳戶中的所有 AWS 服務 和資源。作為安全最佳實務，請將管理存取權指派給使用者，並且僅使用根使用者來執行需要根使用者存取權的任務。

AWS 註冊過程完成後，會向您發送確認電子郵件。您可以隨時前往 https://aws.amazon.com/並選擇「我的帳戶」，檢視目前的帳戶活動並管理您的帳戶。

建立具有管理存取權的使用者

註冊後，請保護您的 AWS 帳戶 AWS 帳戶根使用者 AWS IAM Identity Center、啟用和建立系統管理使用者，這樣您就不會將 root 使用者用於日常工作。

保護您的 AWS 帳戶根使用者

1. 選擇 Root 使用者並輸入您的 AWS 帳戶 電子郵件地址，以帳戶擁有者身分登入。AWS Management Console在下一頁中，輸入您的密碼。

如需使用根使用者登入的說明，請參閱 AWS 登入 使用者指南中的以根使用者身分登入。

2. 為您的 root 使用者開啟多因素驗證 (MFA)。

   如需指示，請參閱《使用指南》中的「IAM為 AWS 帳戶 root 使用者啟用虛擬MFA裝置 (主控台)」。

建立具有管理存取權的使用者

1. 啟用IAM身分識別中心。

   如需指示，請參閱 AWS IAM Identity Center 使用者指南中的啟用 AWS IAM Identity Center。

2. 在IAM身分識別中心中，將管理存取權授與使用者。

   若要取得有關使用 IAM Identity Center 目錄 做為身分識別來源的自學課程，請參閱《使用指南》IAM Identity Center 目錄中的「以預設值設定使用AWS IAM Identity Center 者存取」。

以具有管理存取權的使用者身分登入

- 若要使用您的 IAM Identity Center 使用者登入URL，請使用建立IAM身分識別中心使用者時傳送至您電子郵件地址的登入資訊。

  如需使用IAM身分識別中心使用者登入的說明，請參閱使用指南中的登入 AWS 存取入口網站。AWS 登入

指派存取權給其他使用者

1. 在 IAM Identity Center 中，建立遵循套用最低權限權限的最佳作法的權限集。

   如需指示，請參閱《AWS IAM Identity Center 使用者指南》中的建立許可集。

2. 將使用者指派至群組，然後對該群組指派單一登入存取權。

   如需指示，請參閱《AWS IAM Identity Center 使用者指南》中的新增群組。

AWS 大型主機現代化概念

AWS 大型主機現代化提供工具和資源，協助您在上移轉、現代化及執行大型主機工作負載。 AWS您可以使用此頁面來瞭解大型 AWS 主機現代化中的各種概念，包括應用程式、現代化、環境、重構、重構和執行階段引擎。

主題

- 應用程式
- 應用定義
- Batch 工作
- 組態
- 資料集
- 環境
- 大型主機現代化
- 移民旅程
- 掛載點
- 自動重構
- 平台轉換
- 資源
- 运行时引擎

應用程式

大型主機現 AWS 代化中執行中的大型主機工作負載。一組批次工作、互動式交易 (CICS或IMS) 或其他元件構成應用程式。您可以定義範圍。您必須定義並指定工作負載所需的任何元件或資源，例如CICS交易或批次工作。

應用定義

在 AWS 大型主機現代化中執行的應用程式 (大型主機工作負載) 所需元件和資源的定義或規格。將定義與應用程式本身分開來很重要，因為可以針對由不同執行階段環境所表示的多個階段 (前置生產、生產環境) 重複使用相同的定義。

Batch 工作

設定為不需要使用者互動即可執行的排程程式。在 AWS 大型主機現代化中，您需要將批次任務JCL檔案和批次任務二進位檔案同時存放在 Amazon S3 儲存貯體中，並在應用程式定義檔案中提供兩者的位置。當您執行批次工作時， AWS 大型主機現代化會報告下列狀態值：

提交

批次工作正在提交中。

保留

批次工作處於保留狀態。

派遣

批次工作正在傳送的程序中。

執行中

批次工作目前正在執行中。

取消

批次工作正在取消中。

已取消

批次工作已取消。

Succeeded

批次工作已成功執行。

失敗

批次工作失敗。

已成功但有警告

批次工作成功執行完成，並報告了一個小錯誤。作為 GetBatchJobExecution 響應一部分返回的工作條件代碼表示錯誤的原因。

組態

環境或應用程式的特性。環境組態包括引擎類型、引擎版本、可用性模式、選用的檔案系統組態等。

應用程序配置可以是靜態或動態的。只有當您透過部署新版本來更新應用程式時，靜態組態才會變更。動態組態 (通常是作業活動 (例如開啟或關閉追蹤) 會在您更新時立即變更。

資料集

包含供應用程式使用之資料的檔案。

環境

AWS 計算資源、執行階段引擎和組態詳細資料的具名組合，以代管一或多個應用程式而建立。

大型主機現代化

將應用程式從舊式大型主機環境移轉至. AWS

移民旅程

遷移和現代化舊版應用 end-to-end 程式的程序，通常由下列階段組成：評估、動員、遷移和現代化，以及操作和最佳化。

掛載點

檔案系統中的一個目錄，可讓您存取儲存在該系統中的檔案。

自動重構

將舊版應用程式構件現代化，以便在現代雲端環境中執行的程序。它可以包括代碼和數據轉換。如需詳細資訊，請參閱AWS 大型主機現代化自動重構。

平台轉換

將應用程式和應用程式構件從一個運算平台移至不同運算平台的程序。如需詳細資訊，請參閱AWS 大型主機現代化重新平台。

資源

計算機系統中的物理或虛擬組件。

运行时引擎

促進應用程序運行的軟件。

現代化方法

遷移是複雜的，有許多變量。 AWS 大型主機現代化提供了一種進化方法，通過提高靈活性，並提供大量機會以後優化和創新，從而提供了一些短期的勝利。此外， AWS 大型主機現代化有助於簡化旅程，並且仍然尊重客戶公司和業務的詳細資訊。 AWS 大型主機現代化支援的兩個主要方法是自動重構或重新平台。選擇哪一個取決於您的客戶的情況。

自動重構使用 AWS Blu Age 工具自動將代碼，數據和依賴關係轉換為現代語言，數據存儲和框架，同時保證具有相同業務功能的功能等效性。

重新平台使用 Micro Focus 工具將大型主機工作負載轉換為上的敏捷服務。 AWS

您可以分階段思考現代化的旅程。第一階段包括三個階段：評估、動員和遷移和現代化。下一個階段包括操作和優化階段，您可以在其中找出更多創新機會。

主題

- 評估階段
- 動員階段
- 遷移和現代化階段
- 操作和優化相位

評估階段

在最高層級中，「評估」階段會查看您是否已準備好移轉。您定義一個商業案例，然後通過研討會和沉浸式的一天（演示和實驗室）來教育您的團隊 AWS。研討會和沉浸日解決不同的主題。這些工作是在 AWS 大型主機現代化之外進行的。

動員階段

在 Mobilize 階段，您可以啟動專案，然後執行探索程序，從您的大型主機應用程式擷取資料並將其導入移轉工具。您可以識別要移轉的應用程式，然後選取一些要試驗的應用程式。您可以調整商業案例、撰寫移轉計劃，並決定要如何處理安全性與合規性、帳戶治理及營運模式。您可以與團隊中合適的人員一起建立卓越的雲端中心。你運行飛行員並記錄你學到的東西。您可以優化移轉計劃和商業案例。其中許多工作都是在 AWS 大型主機現代化之外進行的。

遷移和現代化階段

「遷移和現代化」階段適用於每個應用程式，包括指派人員、執行深入探索、找出正確的應用程式架構 AWS、設定應用程式執行階段環境、重新組織或重構程式碼、與其他系統整合，當然還有測試。在階段結束時，您可以將重新整理或重構的應用程式部署到生產環境中，然後切換到新的系統上。AWS這些工作大部分或全部都是在大 AWS 型主機現代化、其他 AWS 服務或大型 AWS 主機現代化提供存取權的工具中執行。

如果您想使用自動重構，請參閱 Blu 見解。 AWS 藍光見解現在可 AWS Management Console 透過單一登入取得。您不必再管理單獨的 AWS 藍光見解憑據。您可以直接從訪問 AWS AWS 藍光時代代碼庫和轉型中心功能。 AWS Management Console

若要將資料從大型主機移轉到 AWS，我們建議使用 AWS SCT 和. AWS Database Migration Service 如需詳細資訊，請參閱什麼是AWS結 Schema Conversion Tool？ 「AWS Schema Conversion Tool 使用指南」中的「AWSDatabase Migration Service」是什麼？ 在《AWS Database Migration Service 使用者指南》中。

操作和優化相位

在「操作和最佳化」階段中，您可以專注於監控已部署的應用程式、管理資源，以及確保安全性和合規性是最新的。您還可以評估將移轉的工作負載最佳化的機會。

開始使用 AWS 大型主機現代化

您可以透過下列向您介紹服務和每個執行階段引擎的教學課程，開始使用 AWS 大型主機現代化。

主題

- 教程：為 AWS 藍光時代設置託管運行時
- 教學課程：設定 Micro Focus 的受管理執行階段

若要繼續學習，請參閱下列自學課程。

- 教學課程：為 BankDemo 範例應用程式設定 Micro Focus 組建
- 教學課程：設定 CI/CD 管道，以便與微焦點企業開發人員搭配使用

教程：為 AWS 藍光時代設置託管運行時

您可以使用本教學課程中指定的示範應用程式，將 AWS Blu AWS Age 現代化應用程式部署到大型主機現代化執行階段環境中。

主題

- 必要條件
- 步驟 1：上傳演示應用程序
- 步驟 2：建立應用程式定義
- 步驟 3：建立執行階段環境
- 步驟 4：建立應用程式
- 步驟 5：部署應用程式
- 步驟 6：啟動應用程式
- 步驟 7：訪問應用程序
- 步驟 8：測試應用程式
- 清除資源

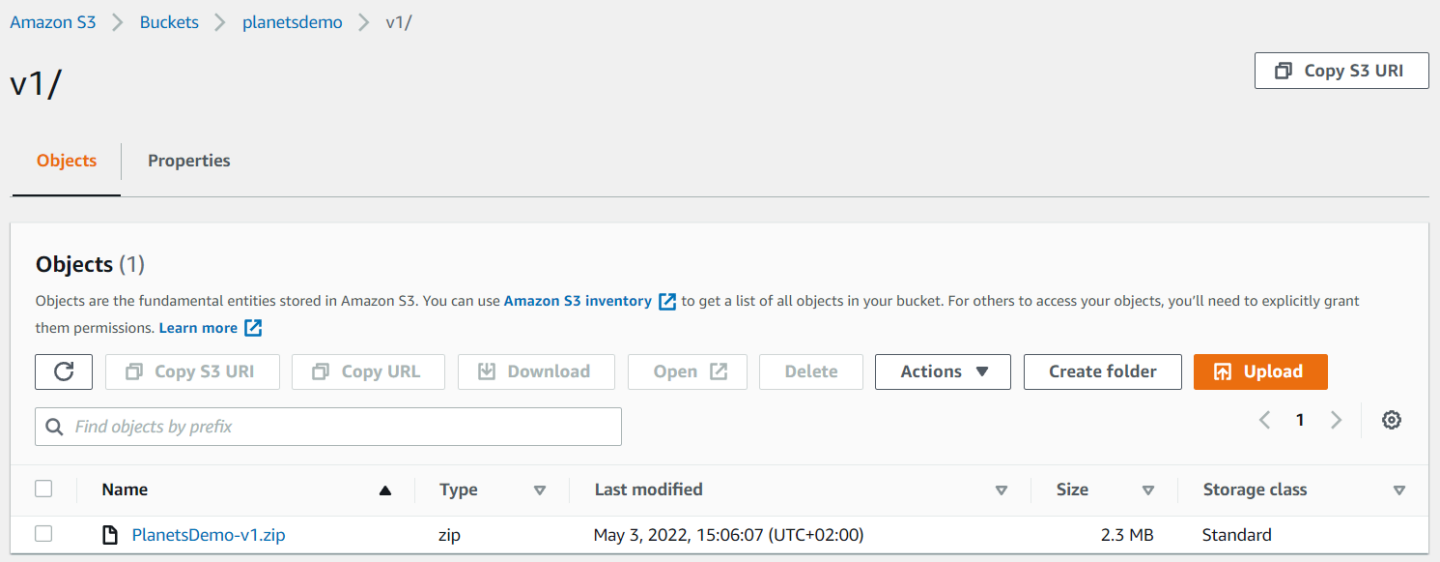
必要條件

要完成本教程，請下載演示應用程序歸檔 PlanetsDemo-v1.zip。

正在運行的演示應用程序需要一個新的瀏覽器訪問。無論您是從桌面還是從 Amazon 彈性運算雲端執行個體執行此瀏覽器 (例如，在中) 都會決定您的安全設定。VPC

步驟 1：上傳演示應用程序

將示範應用程式上傳到 Amazon S3 儲存貯體。請確定此值區與您要部署應用程式的 AWS 區域 位置相同。下列範例顯示名為 Plansdemo 的值區，其中含有 key prefix 或資料夾 (名為 v1)，以及名為的歸檔。`planetsdemo-v1.zip`



> **Note**
>
> 值區中的資料夾為必要資料夾。

步驟 2：建立應用程式定義

若要將應用程式部署到受管理的執行階段，您需要 AWS 大型主機現代化應用程式定義。此定義是JSON描述應用程式位置和設定的檔案。下面的例子是這樣的演示應用程序定義的應用程序：

```
{
    "template-version": "2.0",
    "source-locations": [{
        "source-id": "s3-source",
        "source-type": "s3",
        "properties": {
            "s3-bucket": "planetsdemo",
```

```
            "s3-key-prefix": "v1"
        }
    }],
    "definition": {
        "listeners": [{
            "port": 8196,
            "type": "http"
        }],
        "ba-application": {
            "app-location": "${s3-source}/PlanetsDemo-v1.zip"
        }
    }
}
```

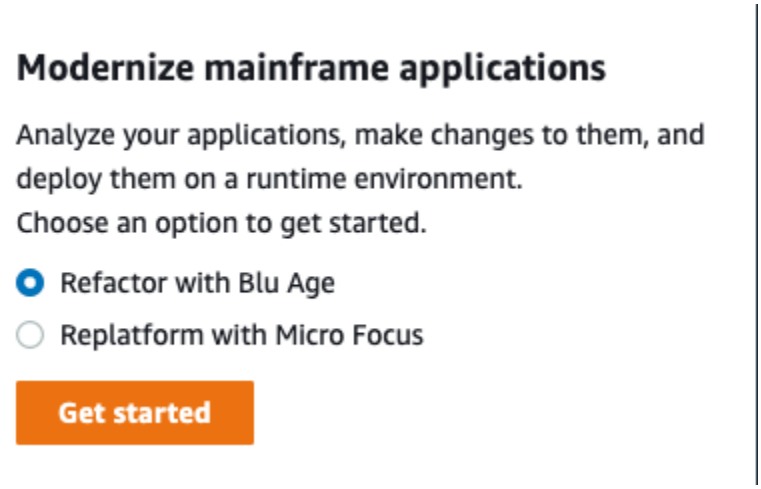
將s3-bucket項目變更為儲存範例應用程式 zip 檔案的值區名稱。

如需應用程式定義的詳細資訊，請參閱AWS 藍光時代的應用程序定義。

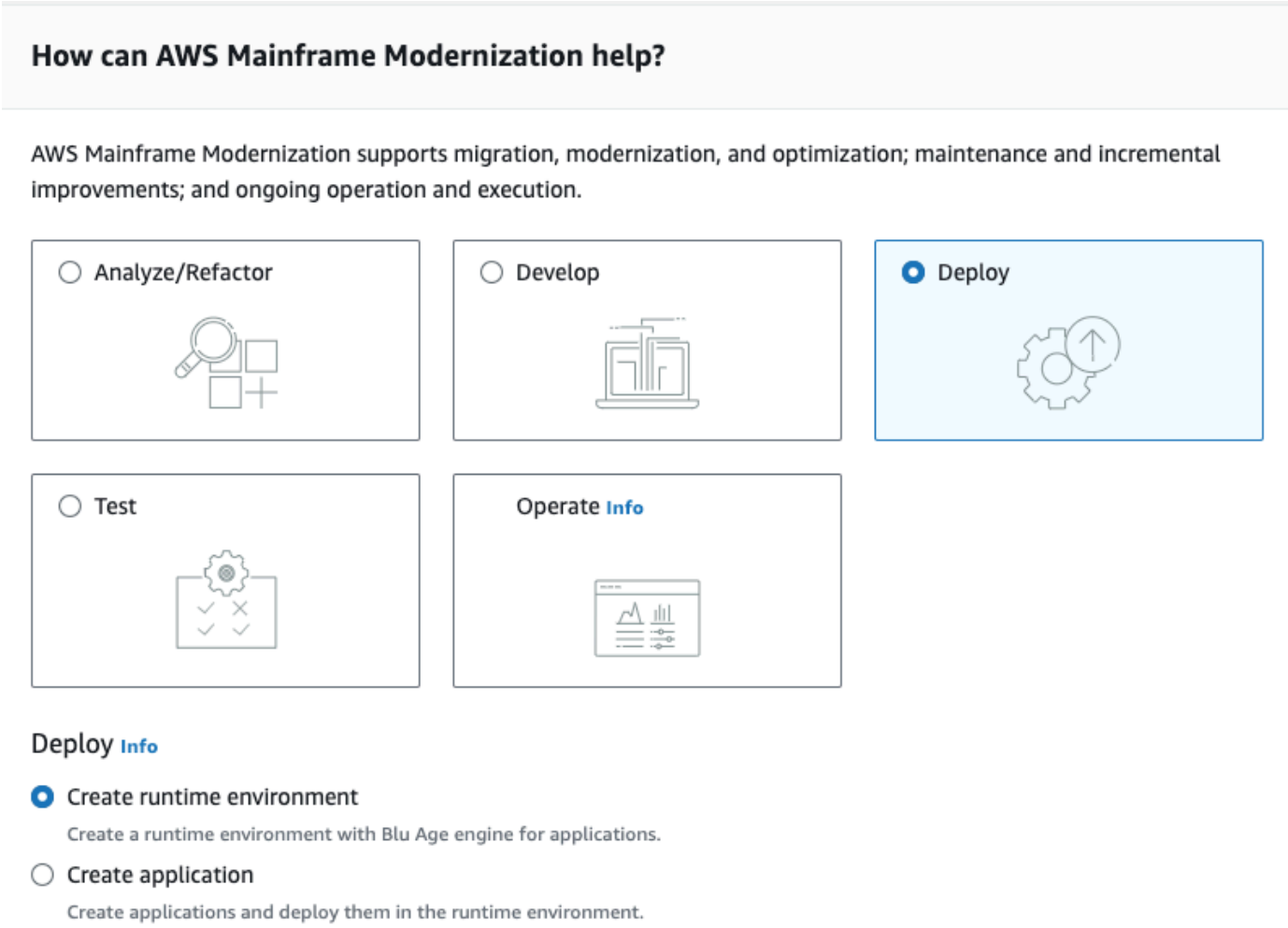
步驟 3：建立執行階段環境

若要建立 AWS 大型主機現代化執行階段環境，請執行下列步驟：

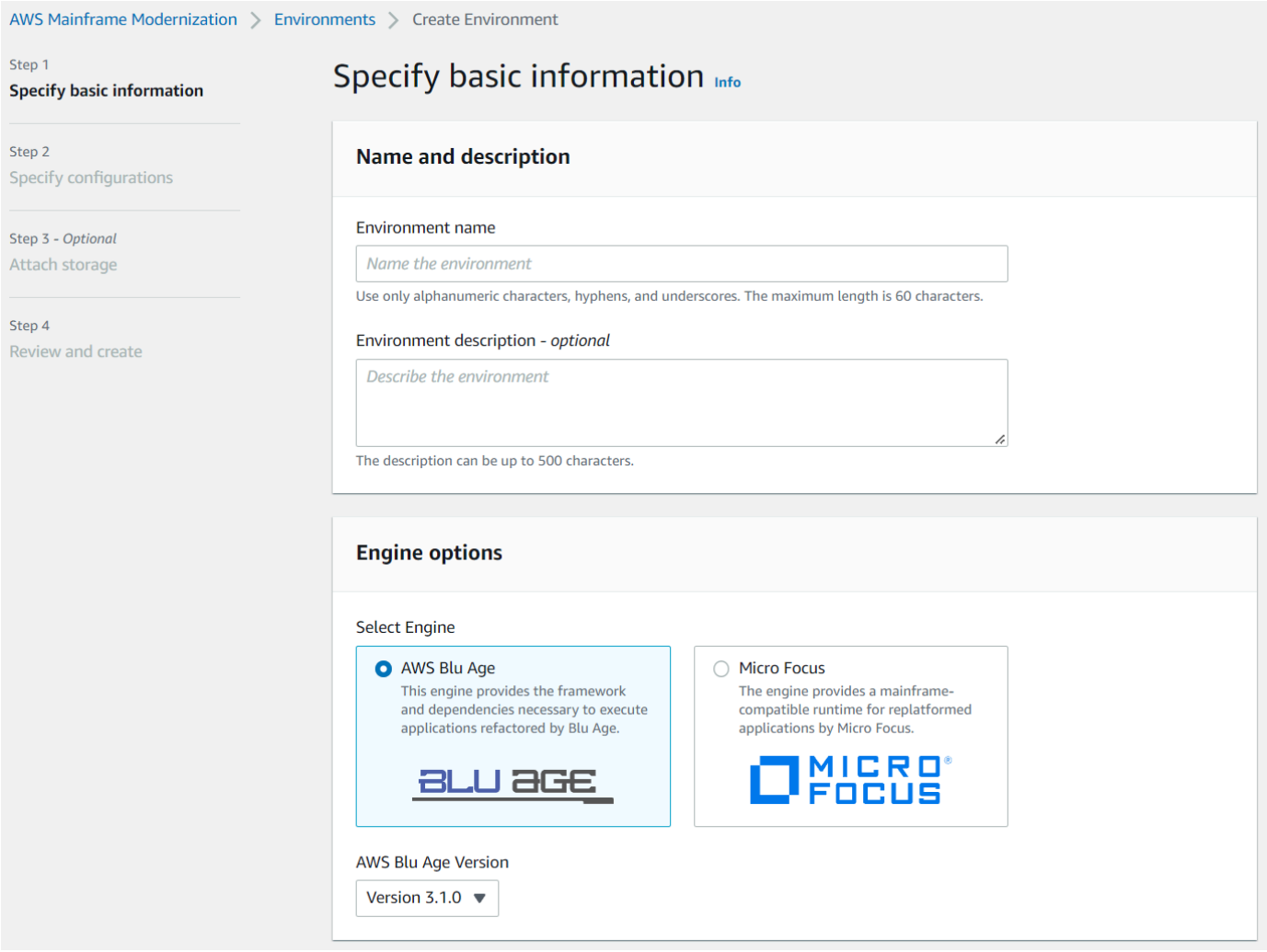
1. 開啟大型主AWS 機現代化主控台。
2. 在選取 AWS 區域 器中，選擇您要建立環境的「區域」。這 AWS 區域 必須與您在中建立 S3 儲存貯體的區域相符步驟 1：上傳演示應用程序。
3. 在「現代化大型主機應用程式」下，選擇「使用 Blu Age 重構」，然後選擇「開始使用」。



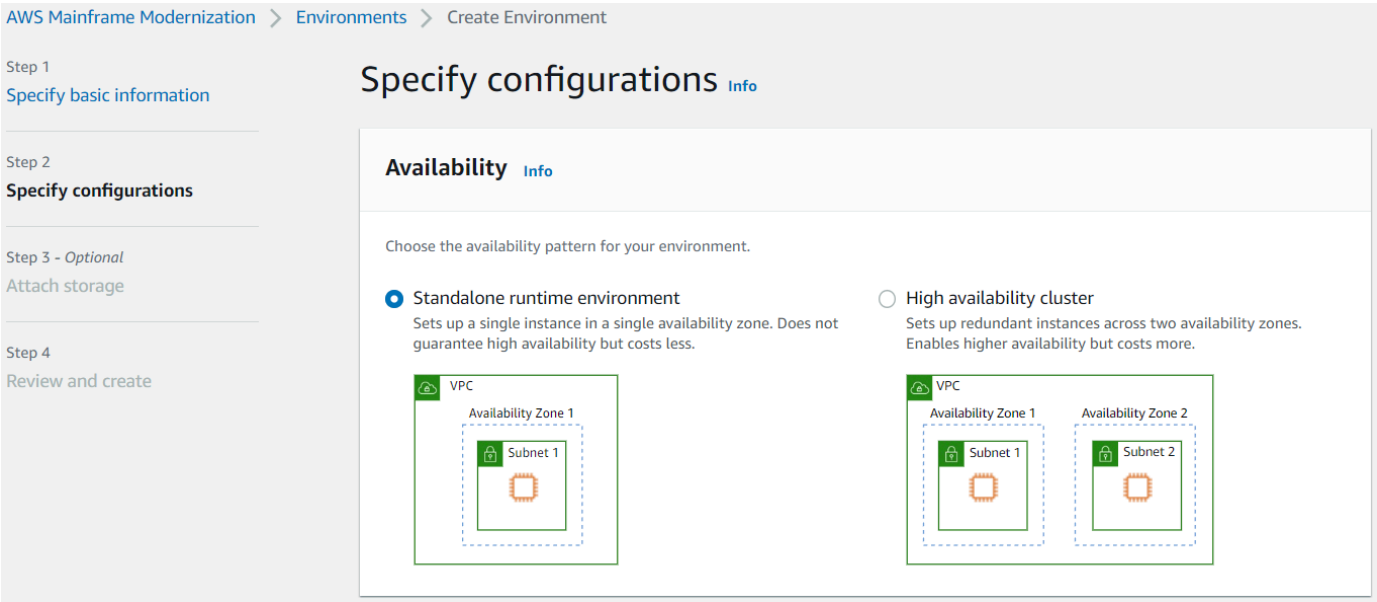
4. 在「AWS大型主機現代化如何協助」下，選擇「部署和建立執行階段環境」。



5. 在左側導覽中，選擇「環境」，然後選擇「建立環境」。在 [指定基本資訊] 頁面上，輸入環境的名稱和說明，然後確定已選取 AWSBlu Age 引擎。或者，您可以將標籤新增至建立的資源。然後選擇下一步。

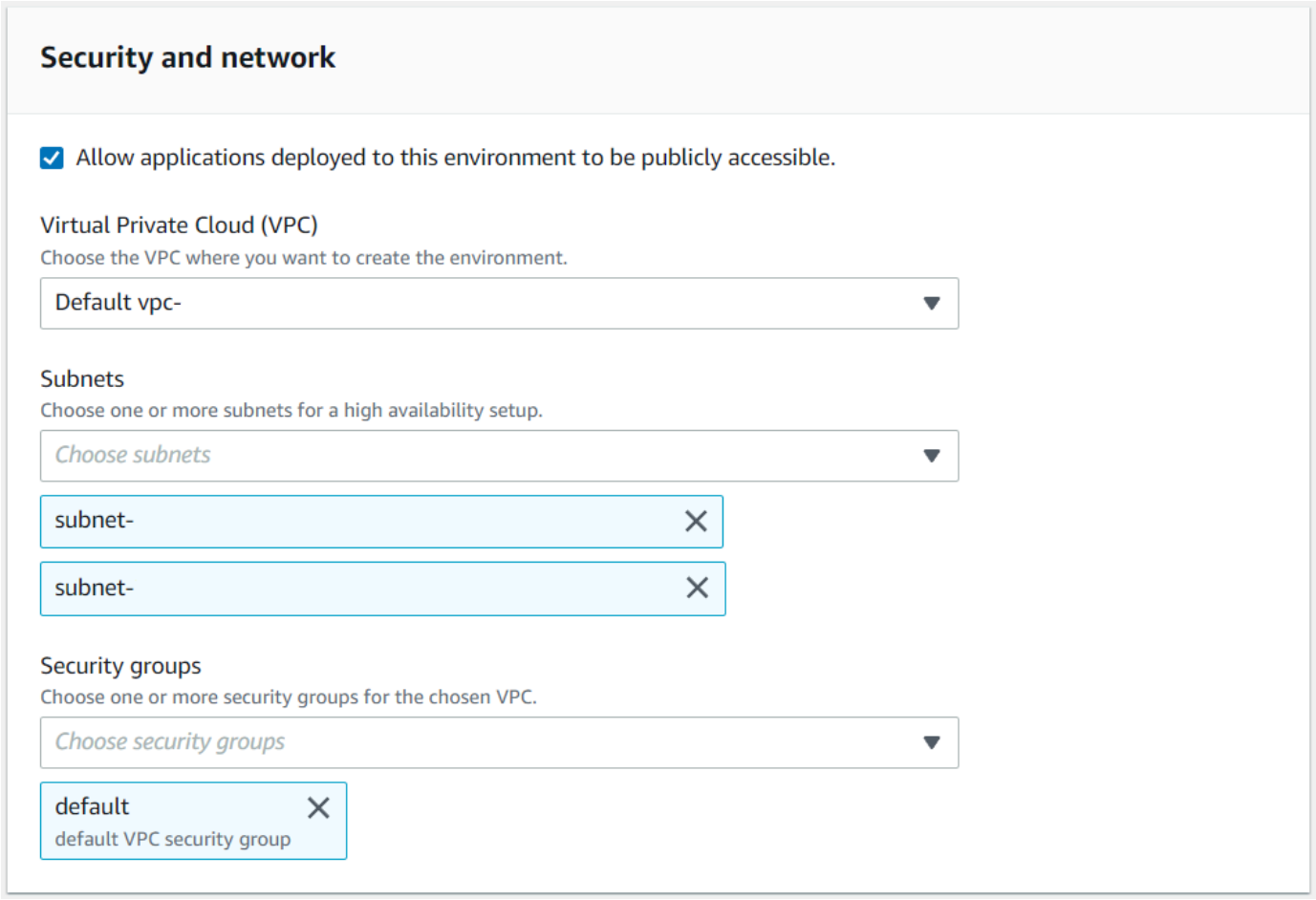


6. 在 [指定組態] 頁面上，選擇 [獨立執行環境]。



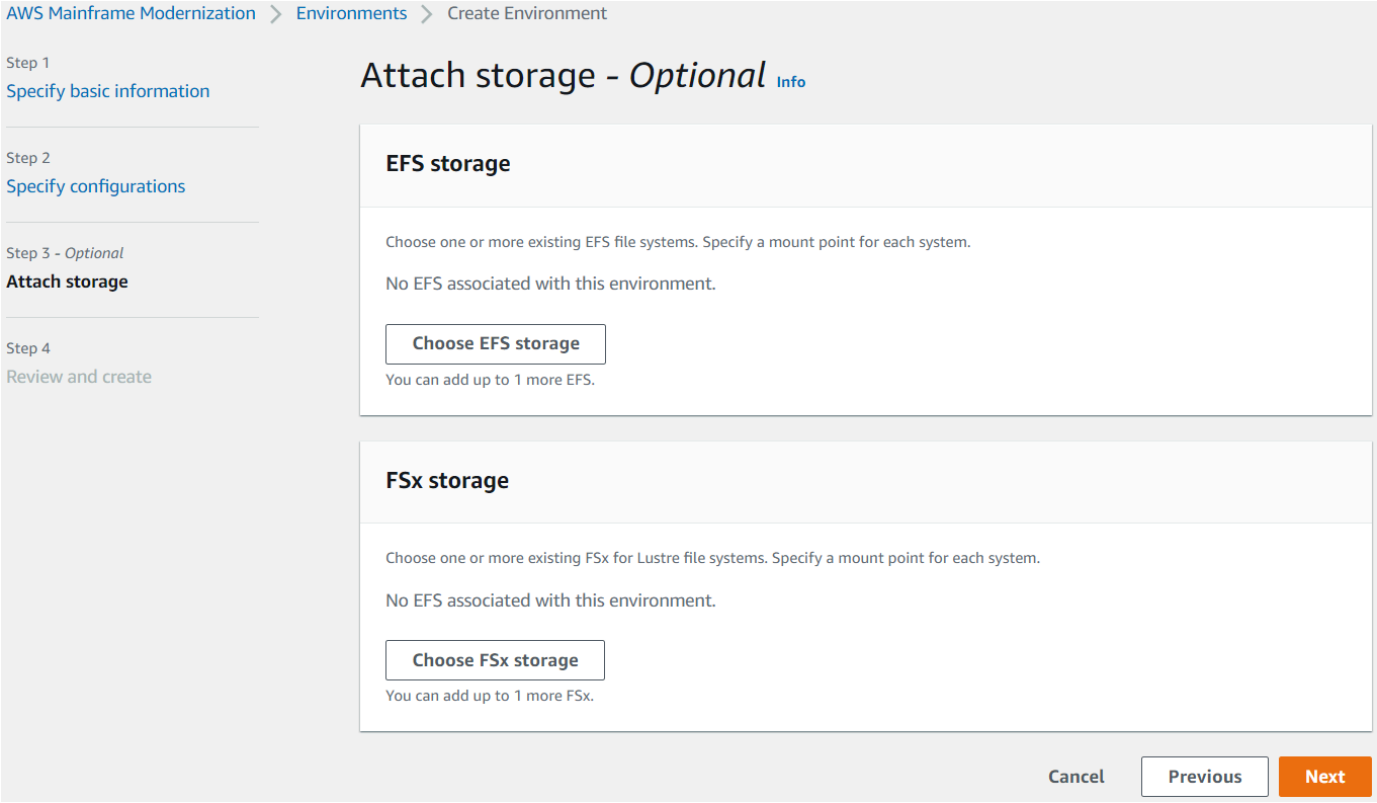
7. 在 [安全性與網路] 底下，進行下列變更：

   - 選擇「允許部署到此環境的應用程式可公開存取」。此選項會將公用 IP 位址指派給應用程式，以便您可以從桌面存取該位址。
   - 選擇一個VPC。您可以使用預設值。
   - 選擇兩個子網路。請確定子網路允許指派公用 IP 位址。
   - 選擇安全群組。您可以使用預設值。請確定您選擇的安全性群組允許從瀏覽器 IP 位址存取您在應用程式定義內`listener`容中指定的連接埠。如需詳細資訊，請參閱步驟 2：建立應用程式定義。



   如果您想要從選擇的外部存取應用程式，請VPC確定已正確設定該應用程式VPC的輸入規則。如需詳細資訊，請參閱疑難排解錯誤：無法存取應用程式 URL。

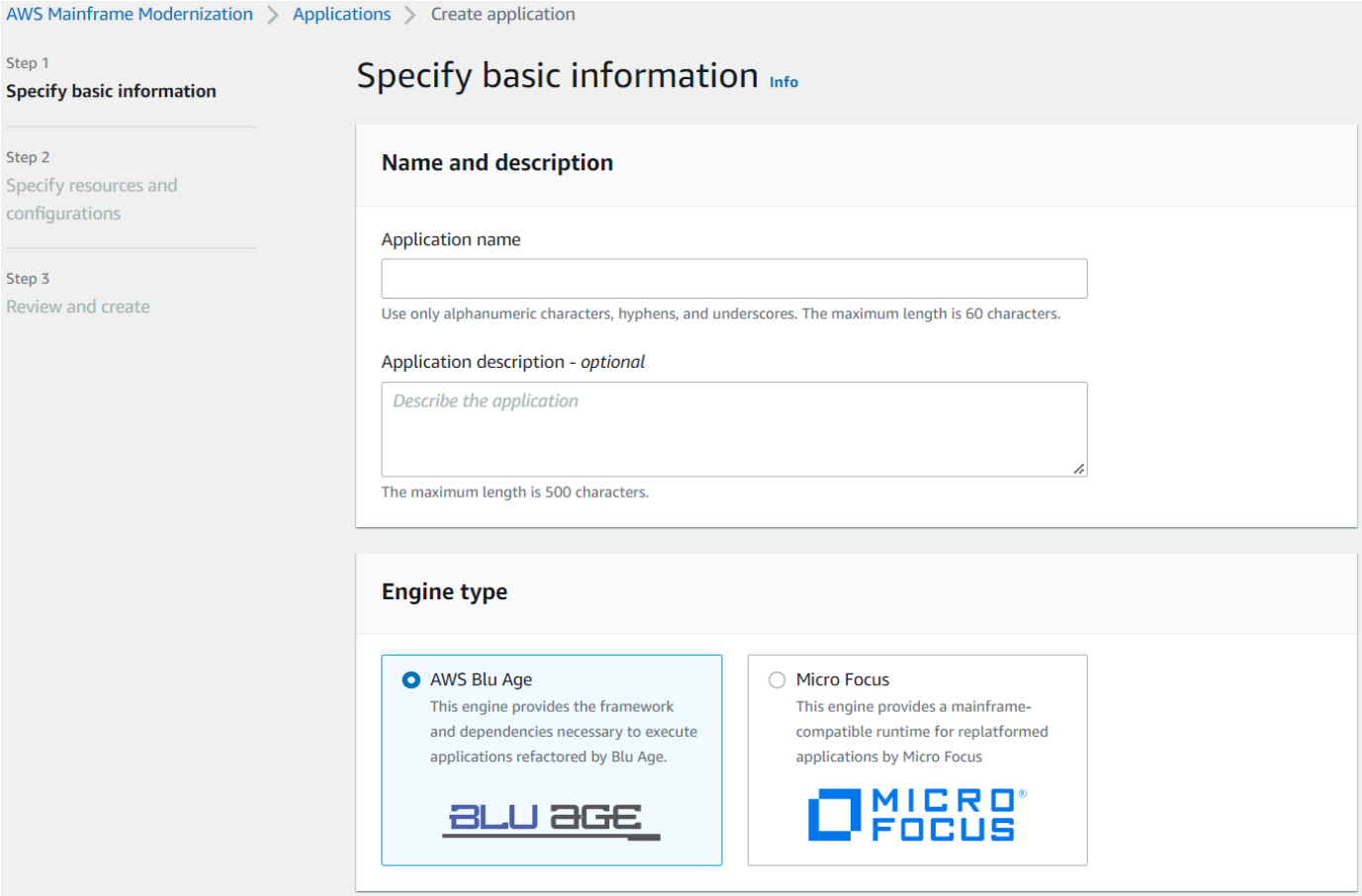
8. 選擇 Next (下一步)。
9. 在附加儲存空間-選用中，保留預設選項並選擇下一步。



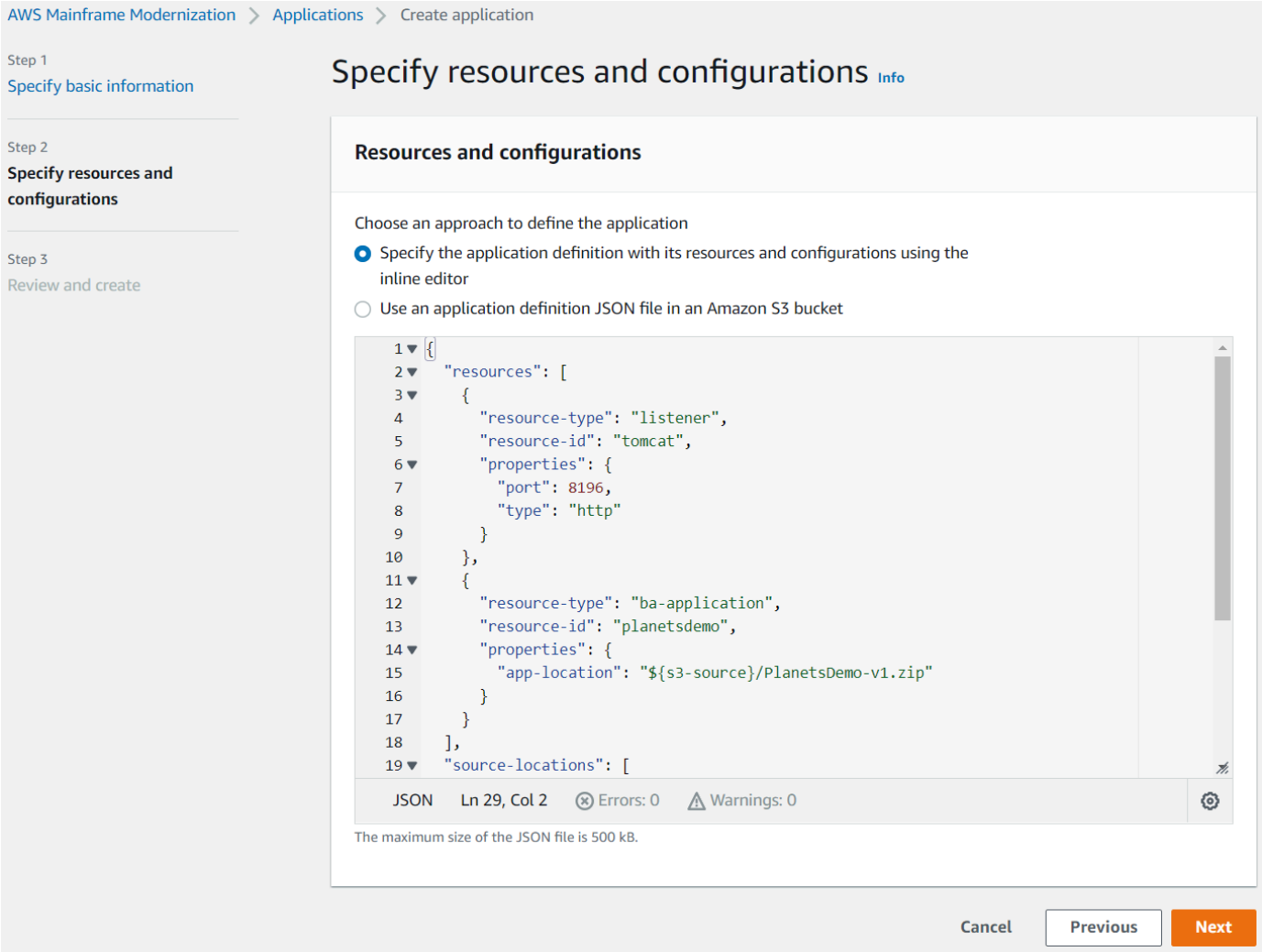
10. 在 [排程維護] 中，選擇 [無偏好設定]，然後選擇 [下一步
11. 在 [檢閱並建立] 中檢閱資訊，然後選擇 [建立環境]。

步驟 4：建立應用程式

1. 導覽至中的AWS大型主機現代化。 AWS Management Console
2. 在導覽窗格中，選擇 Applications (應用程式)，然後選擇 Create application (建立應用程式)。在 [指定基本資訊] 頁面上，輸入應用程式的名稱和說明，並確定已選取 AWSBlu Age 引擎。然後選擇下一步。



3. 在 [指定資源和組態] 頁面上，複製並貼上JSON您在中建立的更新應用程式定義the section called "步驟 2：建立應用程式定義"。

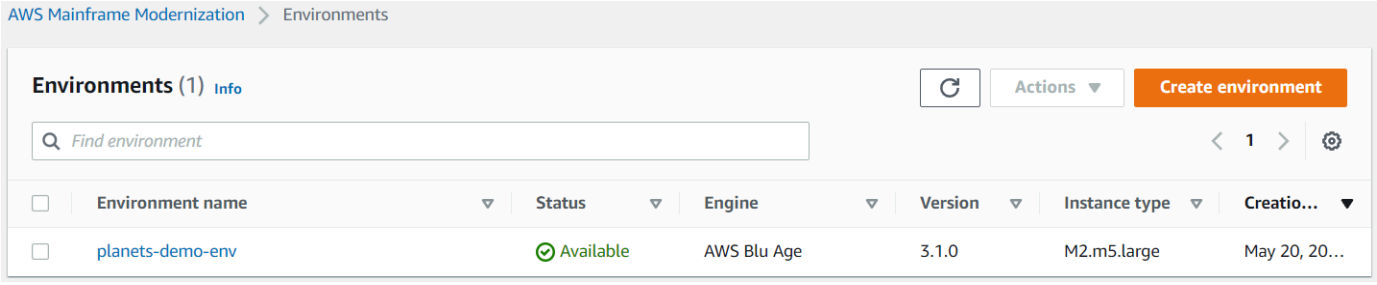


4. 在 [檢閱並建立] 中檢閱您的選擇，然後選擇 [建立應用程式]。

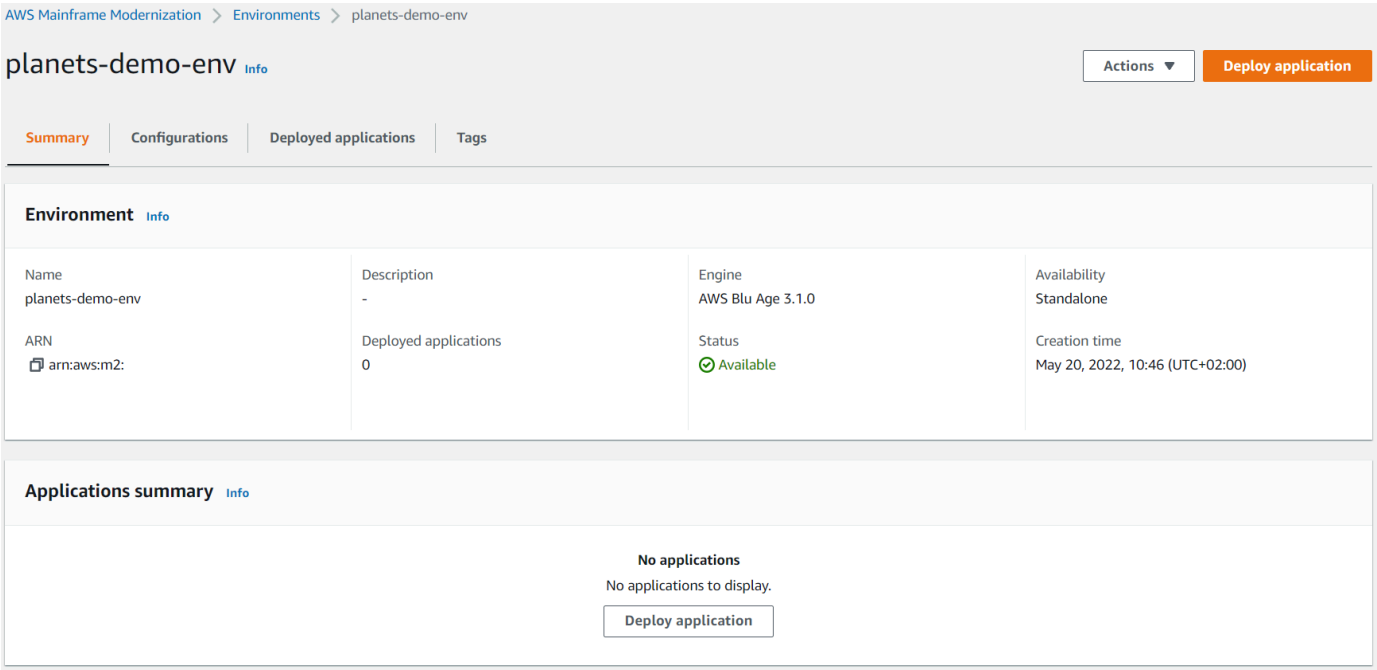
步驟 5：部署應用程式

在您成功建立 AWS 大型主機現代化執行階段環境和應用程式，且兩者都處於 [可用] 狀態之後，您就可以將應用程式部署到執行階段環境中。若要執行此動作，請執行下列步驟。

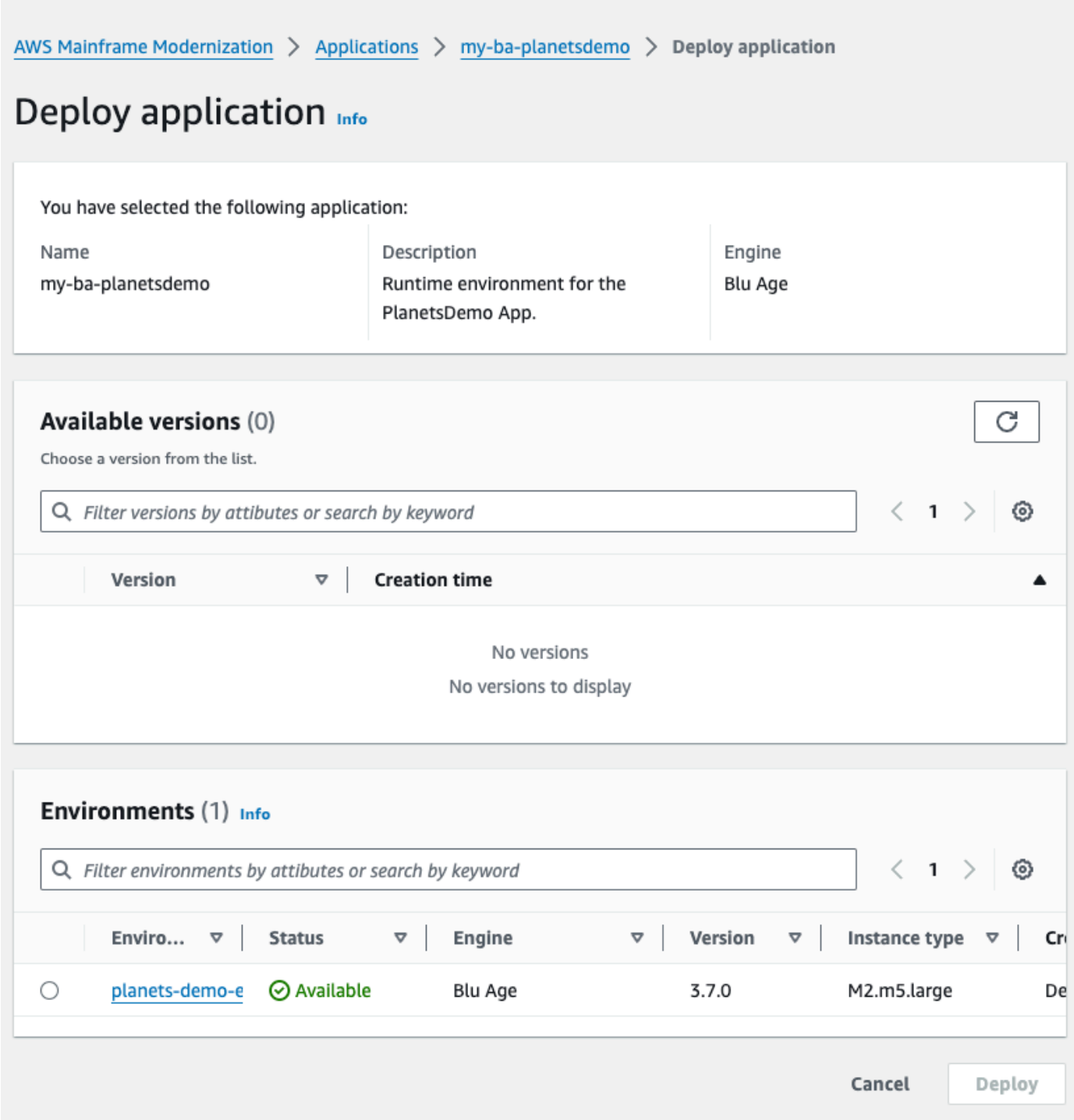
1. 瀏覽至管理AWS主控台中的 AWS 大型主機現代化。在導覽窗格中，選擇 Environments (環境)。接著顯示 [環境] 清單頁面。



2. 選擇先前建立的執行階段環境。隨即顯示「環境詳細資訊」頁面。
3. 選擇部署應用程式。



4. 選擇先前建立的應用程式，然後選擇您要部署應用程式的版本。然後選擇 Deploy (部署)。

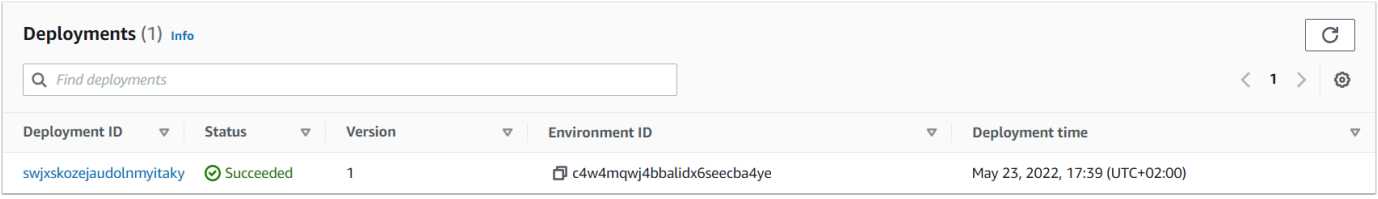


5. 等待應用程式完成部署。您會看到一個橫幅，其中顯示「應用程式已成功部署」訊息。

步驟 6：啟動應用程式

1. 導覽至「AWS大型主機現代化」，AWS Management Console 然後選擇「應用程式」。

2. 選擇您的應用程式，然後移至部署。應用程式的狀態應為 [成功]。



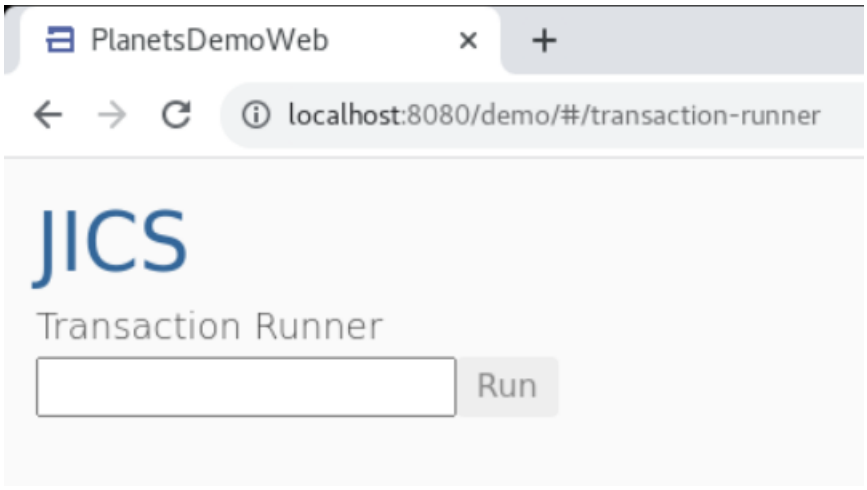
3. 選擇動作，然後選擇啟動應用程式。

步驟 7：訪問應用程序

1. 等待應用程式處於 [執行中] 狀態。您會看到一個橫幅，其中顯示「應用程式已成功啟動」訊息。
2. 複製應用程式DNS主機名稱。您可以在應用程式的 [應用程式資訊] 區段中找到此主機名稱。
3. 在瀏覽器中，導航到`http://{hostname}:{portname}/PlanetsDemo-web-1.0.0/`，其中：

   - `hostname`是先前複製的DNS主機名稱。
   - `portname`是您在中建立的應用程式定義中定義的 Tomcat 連接埠。步驟 2：建立應用程式定義

   JICS螢幕隨即出現。



如果您無法存取應用程式，請參閱疑難排解錯誤：無法存取應用程式 URL。

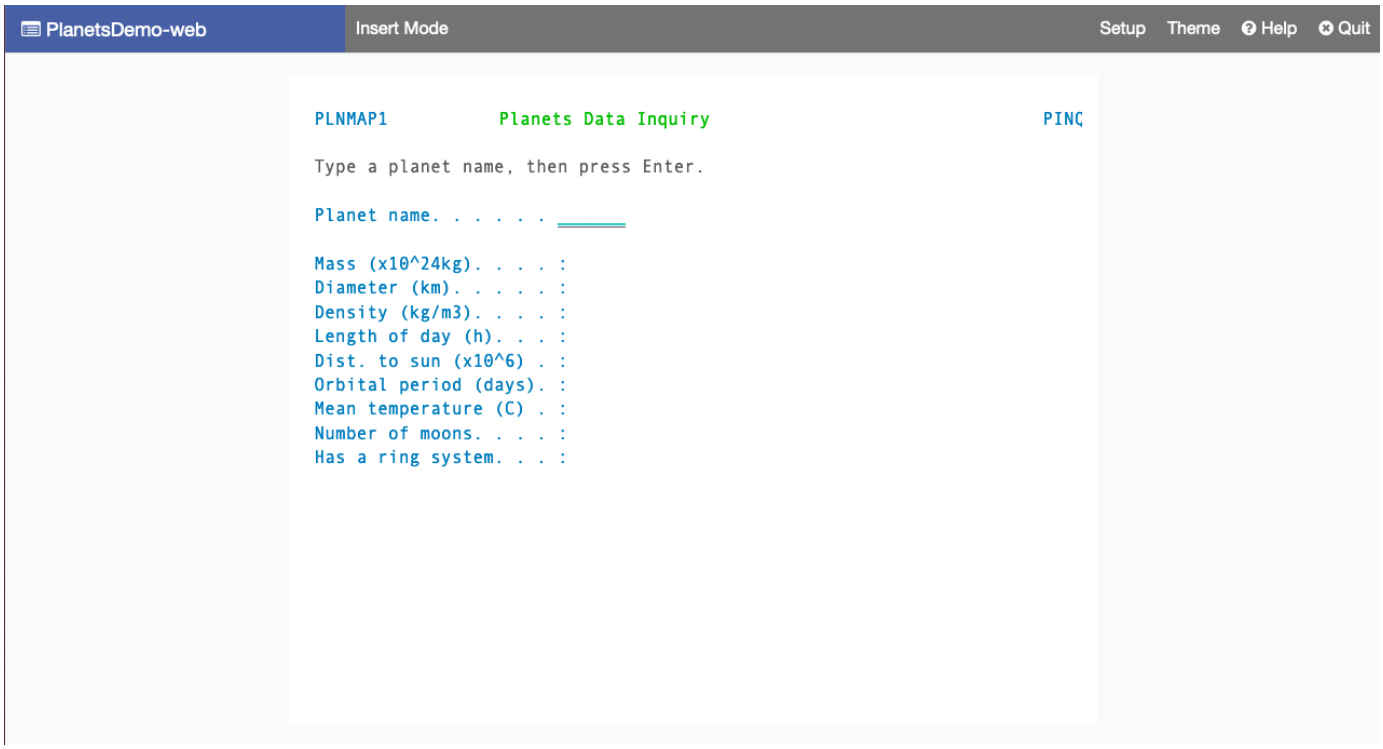
> **Note**
>
> 如果無法存取應用程式，且安全性群組的輸入規則已在連接埠 8196 上選取「我的 IP」，請指定規則以允許連接埠 8196 上的 LB i/p 流量。

步驟 8：測試應用程式

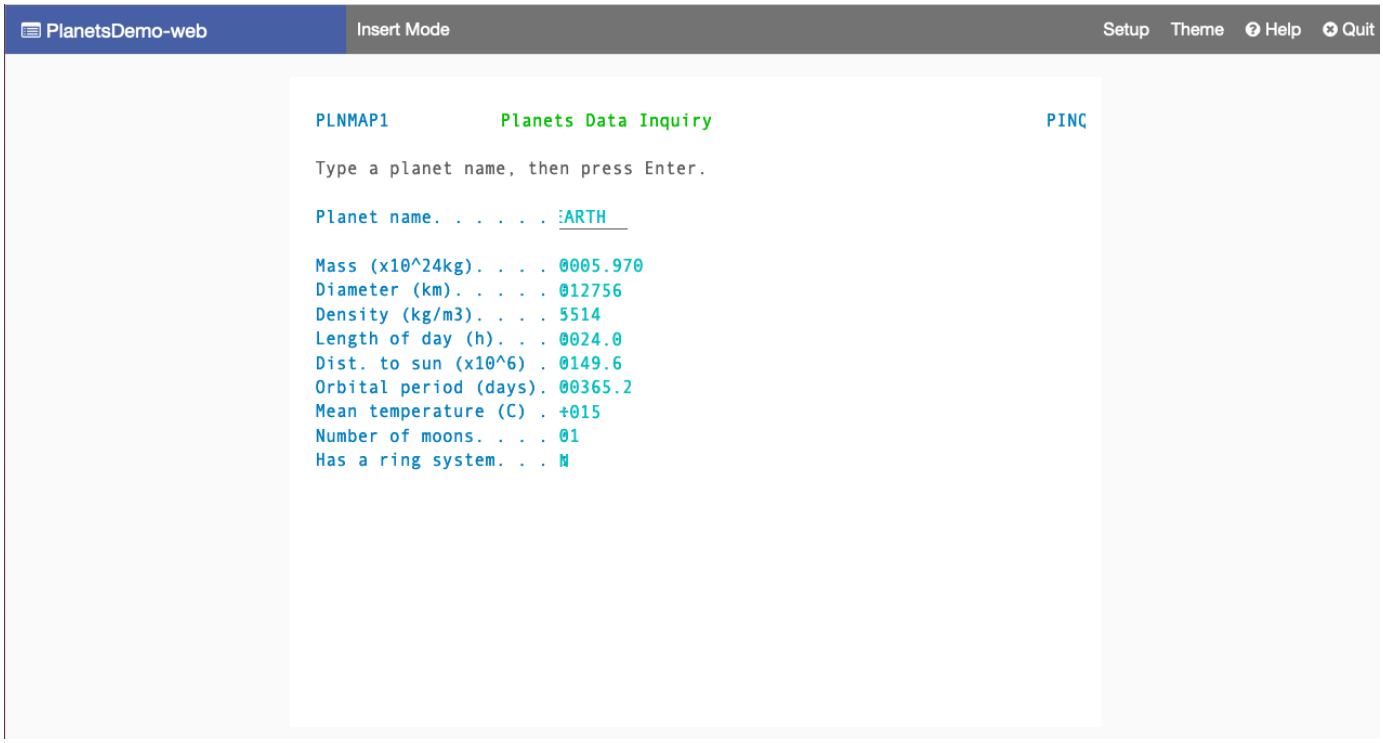
在此步驟中，您會在移轉的應用程式中執行交易。

1. 在JICS畫面上，在輸入欄位PINQ中輸入，然後選擇執行 (或按 Enter) 以啟動應用程式交易。

   示範應該會出現應用程式畫面。



2. 在對應的欄位中輸入行星名稱，然後按 Enter。



你應該看到關於這個星球的細節。

清除資源

如果您不再需要為此教學課程建立的資源，請刪除這些資源以避免額外費用。若要這樣做，請完成下列步驟：

- 如果 AWS 大型主機現代化應用程式仍在執行中，請停止它。
- 刪除 應用程式。如需詳細資訊，請參閱刪除 AWS Mainframe Modernization 應用程式。
- 刪除執行階段環境。如需詳細資訊，請參閱刪除 AWS 大型主機現代化執行階段環境。

教學課程：設定 Micro Focus 的受管理執行階段

您可以使用 Micro Focus 執行階段引擎，在 AWS 大型主機現代化受管理的執行階段環境中部署和執行應用程式。本教學課程 AWS 說明如何使用 Micro Focus 執行階段引擎，在大型主機現代化管理的執行階段環境中部署及執行 CardDemo 範例應用程式。 CardDemo 範例應用程式是一種簡化的信用卡應用程式，專為測試 AWS 和展示大型主機現代化使用案例的技術與合作夥伴而開發。

在教學課程中，您會在其他中建立資源 AWS 服務。其中包括 Amazon 簡單存儲服務，Amazon Relational Database Service AWS Key Management Service，和 AWS Secrets Manager.

主題

- 必要條件
- 步驟 1：建立並載入 Amazon S3 儲存貯體
- 步驟 2：建立和設定資料庫
- 步驟 3：建立並設定 AWS KMS key
- 步驟 4：建立和設定 AWS Secrets Manager 資料庫密碼
- 步驟 5：建立執行階段環境
- 步驟 6：建立應用程式
- 步驟 7：部署應用程式
- 步驟 8：匯入資料集
- 步驟 9：啟動應用程式
- 步驟 10：Connect 到 CardDemo CICS應用程序
- 清除資源
- 後續步驟

必要條件

- 確保您可以訪問 3270 仿真器以使用該CICS連接。免費和試用版 3270 模擬器可從第三方網站獲得。或者，您也可以啟動 AWS 大型主機現代化 AppStream 2.0 Micro Focus 執行個體，並使用 Rumba 3270 模擬器 (不是免費提供)。

  如需 AppStream 2.0 的資訊，請參閱the section called “教學課程：為企業分析器和企業開發人員設定 AppStream 2.0”。

  > **Note**
  >
  > 建立堆疊時，請選擇企業開發人員 (ED) 選項，而不是企業分析器 (EA)。

- 下載CardDemo 範例應用程式，並將下載的檔案解壓縮至任何本機目錄。此目錄將包含一個標題為`CardDemo`的子目錄。
- VPC在您的帳戶中識別一個，您可以在其中定義本教學課程中建立的資源。至少VPC需要兩個可用區域中的子網路。有關 Amazon 的更多信息VPC，請參閱 Amazon 如何VPC工作。

步驟 1：建立並載入 Amazon S3 儲存貯體

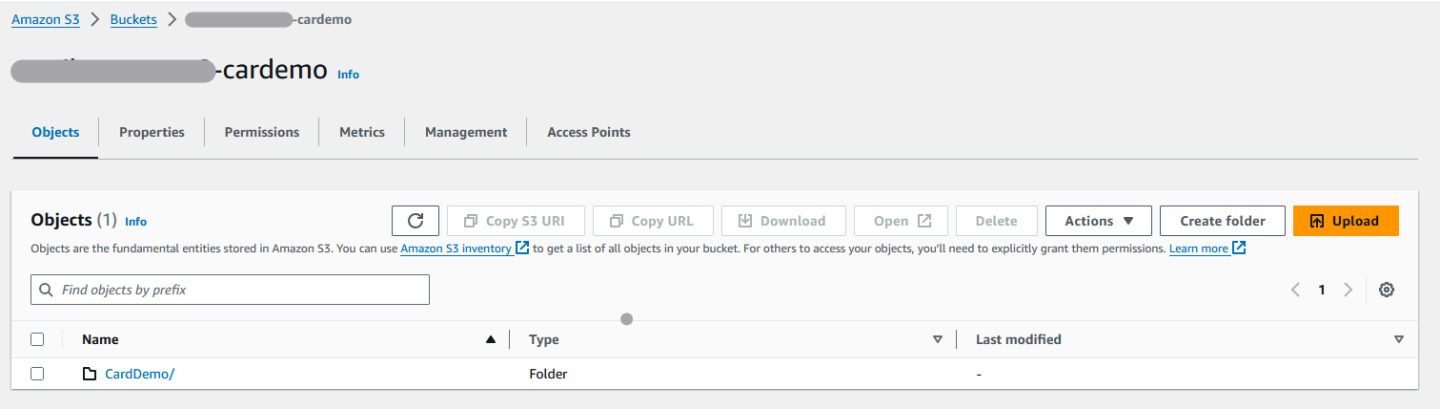
在此步驟中，您會建立 Amazon S3 儲存貯體，並將 CardDemo 檔案上傳到此儲存貯體。稍後在本教學課程中，您會使用這些檔案，在 AWS 大型主機現代化 Micro Focus 受管理執行階段環境中部署和執行 CardDemo 範例應用程式。

> **Note**
>
> 您不需要建立新的 S3 儲存貯體，但您選擇的儲存貯體必須與本教學課程中使用的其他資源位於相同的區域中。

建立 Amazon S3 儲存貯體

1. 開啟 Amazon S3 主控台，然後選擇「建立儲存貯體」。
2. 在 [一般] 組態中，選擇您要建置 AWS 大型主機現代化 Micro Focus 受管理執行階段的AWS區域。
3. 輸入「值區」名稱，例如`yourname-aws-region-carddemo`。保留預設設定，然後選擇 [建立值區]。或者，您也可以從現有的 Amazon S3 儲存貯體複製設定，然後選擇「建立儲存貯體」。
4. 選擇您剛建立的值區，然後選擇 [上傳]。
5. 在 [上傳] 區段中，選擇 [新增資料夾]，然後從您的本機電腦瀏覽至`CardDemo`目錄。
6. 選擇「上傳」以開始上傳程序。上傳時間會根據您的連線速度而有所不同。
7. 上傳完成時，確認所有檔案都已順利上傳，然後選擇 [關閉]。

您的 Amazon S3 存儲桶現在包含該`CardDemo`文件夾。



如需 S3 儲存貯體的相關資訊，請參閱建立、設定和使用 Amazon S3 儲存貯體。

步驟 2：建立和設定資料庫

在此步驟中，您可以在 Amazon 關係SQL數據庫服務（亞馬遜RDS）中創建 Postgre 數據庫。在教學課程中，此資料庫包含 CardDemo 範例應用程式用於有關信用卡交易之客戶作業的資料集。

在 Amazon 中創建數據庫 RDS

1. 打開 Amazon RDS 控制台。
2. 選擇您要在其中建立資料庫執行處理的AWS區域。
3. 在導覽窗格中，選擇 Databases (資料庫)。
4. 選擇 [建立資料庫]，然後選擇 [標準建立]。
5. 針對「引擎類型」，選擇「下一步」。SQL
6. 選擇 15 或更高版本的引擎版本。

> **Note**
>
> 儲存引擎版本，因為您稍後在本教學課程中需要它。

7. 在 範本 區段中，選擇 免費方案。
8. 將資料庫執行個體識別碼變更為有意義的項目，例如`MicroFocus-Tutorial`。
9. 避免在中管理主要認證 AWS Secrets Manager。而是輸入主密碼並進行確認。

> **Note**
>
> 儲存您用於資料庫的使用者名稱和密碼。您將在本教程的後續步驟中安全地存儲它們。

10. 在 [連線] 底下，選擇您VPC要建立 AWS 大型主機現代化管理執行階段環境的位置。
11. 選擇建立資料庫。

在 Amazon 中創建自定義參數組 RDS

1. 在 Amazon 主RDS控台導覽窗格中，選擇「參數群組」，然後選擇「建立參數群組」。
2. 在「建立參數群組」視窗中，對於「參數群組族群」，選取與資料庫版本相符的 Postgres 選項。

> **Note**
>
> 某些 Postgres 版本需要類型。視需要選取「資料庫參數群組」。輸入參數群組的「群組」名稱和「描述」。

3. 選擇 Create (建立)。

規劃自訂參數群組的步驟

1. 選擇新建立的參數群組。
2. 選擇動作，然後選擇編輯。
3. 篩選`max_prepared_transactions`並將參數值變更為 100。
4. 選擇 Save Changes (儲存變更)。

將自訂參數群組與資料庫相關聯的步驟

1. 在 Amazon 主RDS控台導覽窗格中，選擇 [資料庫]，然後選擇要修改的資料庫執行個體。
2. 選擇 Modify (修改)。Modify DB instance (修改資料庫執行個體) 頁面隨即出現。

> **Note**
>
> 在資料庫完成建立和備份之前，無法使用 [修改] 選項，這可能需要幾分鐘的時間。

3. 在 [修改資料庫執行個體] 頁面上，瀏覽至 [其他組態]，然後將資料庫參數群組變更為參數群組。如果清單中沒有您的參數群組，請檢查是否使用正確的資料庫版本建立。
4. 選擇「繼續」，然後檢查修改摘要。
5. 選擇「立即套用」以立即套用變更。
6. 選擇修改資料庫執行個體以儲存您的變更。

如需有關參數群組的詳細資訊，請參閱使用參數群組。

> **Note**
>
> 您也可以將 Amazon Aurora Postgre SQL 資料庫與 AWS 大型主機現代化搭配使用，但沒有免費方案選項。如需詳細資訊，請參閱使用 Amazon Aurora 郵政。SQL

步驟 3：建立並設定 AWS KMS key

若要安全地存放 Amazon RDS 執行個體的登入資料，請先建立 AWS KMS key.

若要建立 AWS KMS key

1. 開啟金鑰管理服務主控台。
2. 選擇 Create Key (建立金鑰)。
3. 保留密鑰類型的對稱默認值，並為密鑰使用加密和解密。
4. 選擇 Next (下一步)。
5. 為金鑰指定別名，例如`MicroFocus-Tutorial-RDS-Key`和可選描述。
6. 選擇 Next (下一步)。
7. 勾選使用者或角色旁邊的核取方塊，以指派金鑰管理員。
8. 選擇 [下一步]，然後再選擇 [下一步]。
9. 在檢閱畫面上，編輯金鑰政策，然後輸入下列內容：

```
{
     "Sid" : "Allow access for Mainframe Modernization Service",
     "Effect" : "Allow",
        "Principal" : {
           "Service" : "m2.amazonaws.com"
                 },
      "Action" : "kms:Decrypt",
      "Resource" : "*"
},
```

   此原則會使用此 AWS 特定金鑰原則授與大型主機現代化解密權限。

10. 選擇「完成」以建立金鑰。

如需詳細資訊，請參閱 AWS Key Management Service 開發人員指南中的建立金鑰。

步驟 4：建立和設定 AWS Secrets Manager 資料庫密碼

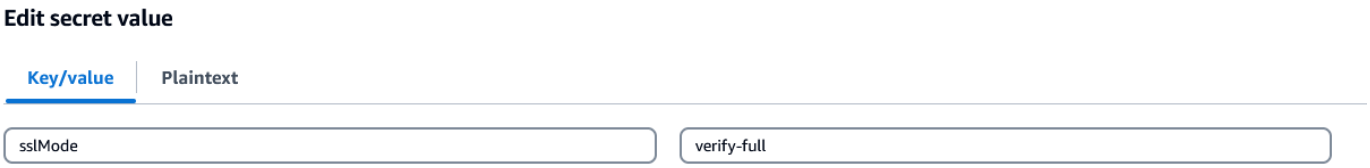
現在，使用 AWS Secrets Manager 和安全地儲存資料庫認證 AWS KMS key。

若要建立和設定資 AWS Secrets Manager 料庫密碼

1. 開啟 Secrets Manager 主控台。
2. 在導覽窗格中，選擇 Secrets (祕密)。
3. 在 [秘密] 中，選擇 [儲存新密碼]。
4. 將密碼類型設置為 Amazon RDS 數據庫的憑據。
5. 輸入您在建立資料庫時指定的「認證」。
6. 在「加密金鑰」下，選取您在步驟 3 中建立的金鑰。
7. 在 [資料庫] 區段中，選取您為此教學課程建立的資料庫，然後選擇 [下一步]。
8. 在密碼名稱下，輸入名稱，例如`MicroFocus-Tutorial-RDS-Secret`和可選描述。
9. 在 [資源權限] 區段中，選擇 [編輯權限]，並以下列原則取代內容：

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect" : "Allow",
            "Principal" : {
                "Service" : "m2.amazonaws.com"
            },
            "Action" : "secretsmanager:GetSecretValue",
            "Resource" : "*"
        }
    ]
}
```

10. 選擇 Save (儲存)。
11. 在後續畫面中選擇 [下一步]，然後選擇 [商店]。重新整理密碼清單以查看新密碼。
12. 選擇新創建的密鑰並記下，`Secret ARN`因為您稍後在教程中需要它。
13. 在密碼的概觀索引標籤中，選擇擷取秘密值。
14. 選擇 [編輯]，然後選擇 [新增列]。
15. 添加一個鍵`sslMode`，其值為`verify-full`：

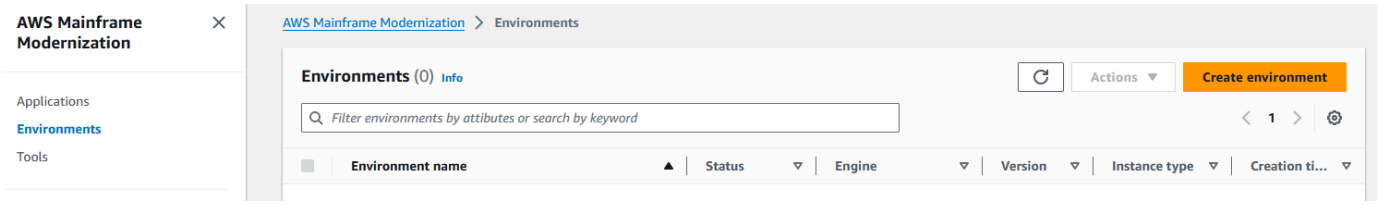


16. 選擇 Save (儲存)。

步驟 5：建立執行階段環境

建立執行階段環境的步驟

1. 開啟大型主AWS 機現代化主控台。
2. 在導覽窗格中，選擇 Environments (環境)。然後選擇建立環境。



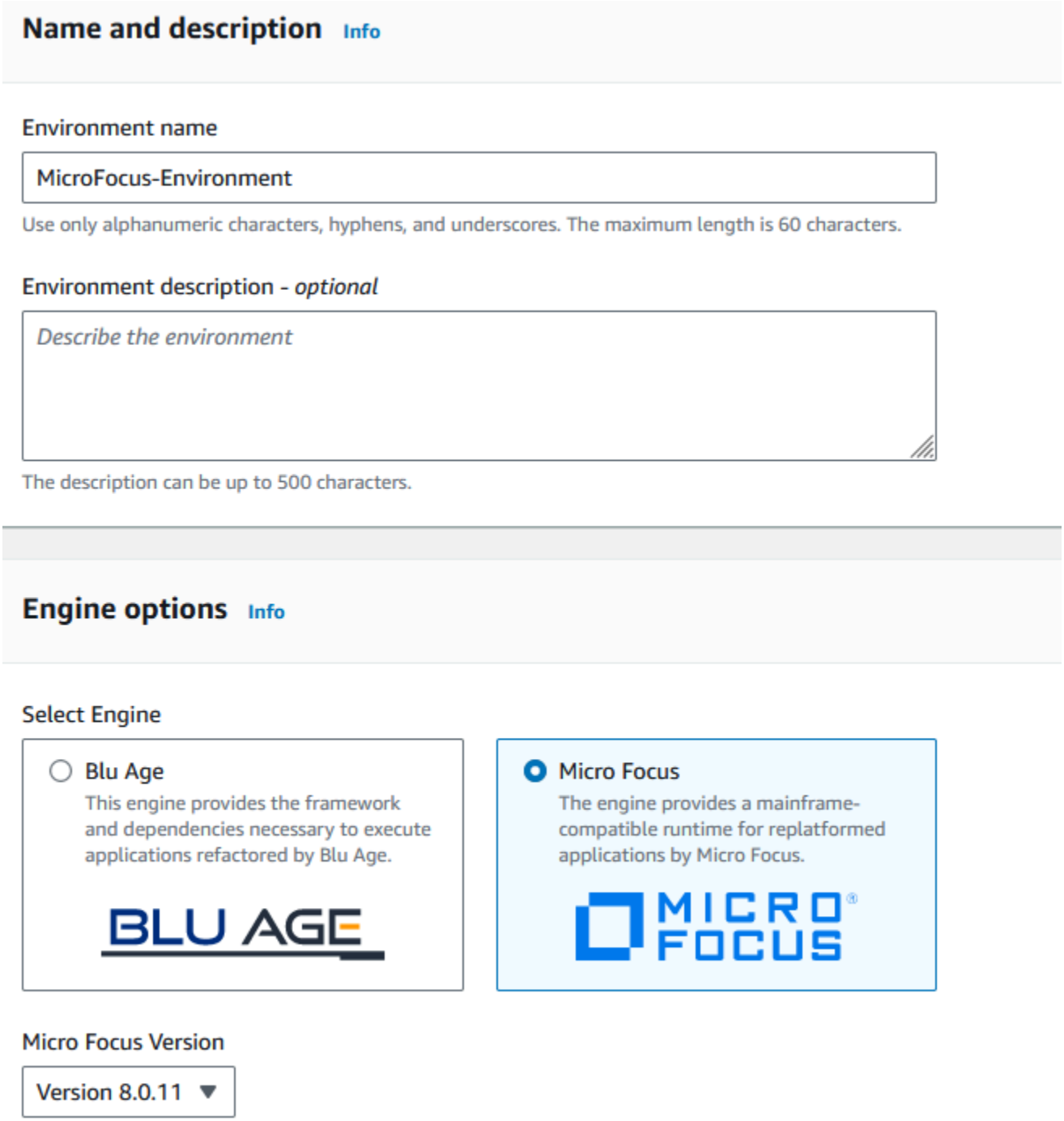
3. 在「指定基本資訊」下，

   a. 輸`MicroFocus-Environment`入環境名稱。

   b. 在「引擎選項」下，確定已選取「微焦點」。

   c. 選擇最新的微焦點版本。

   d. 選擇 Next (下一步)。



4. 設定環境

   a. 在可用性下，選擇高可用性叢集。

   b. 在 [資源] 底下，選擇 [M2.c5.large] 或 [M 2.m5. Large] 做為執行個體類型，以及您想要的執行個體數目。最多可指定兩個例證。

c. 在 [安全性與網路] 底下，選擇 [允許部署到此環境的應用程式可公開存取]，然後選擇至少兩個公用子網路。

d. 選擇 Next (下一步)。

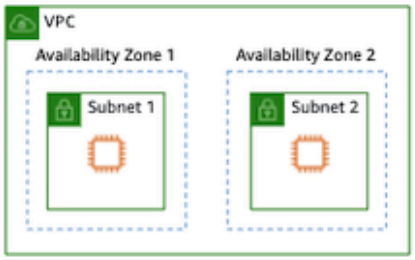
Specify configurations Info

Availability Info

Choose the availability pattern for your environment.

- Standalone runtime environment
  Sets up a single instance in a single availability zone. Does not guarantee high availability but costs less.

- High availability cluster
  Sets up redundant instances across two availability zones. Enables higher availability but costs more.



Resources

Instance type
Choose the instance type for your high availability cluster.

M2.m5.large

Desired capacity
Specify the desired number of instances.

2

Security and network

☑ Allow applications deployed to this environment to be publicly accessible.

Virtual Private Cloud (VPC)
Choose the VPC where you want to create the environment.

Default vpc-15

Subnets
Choose one or more subnets for a high availability setup.

Choose subnets

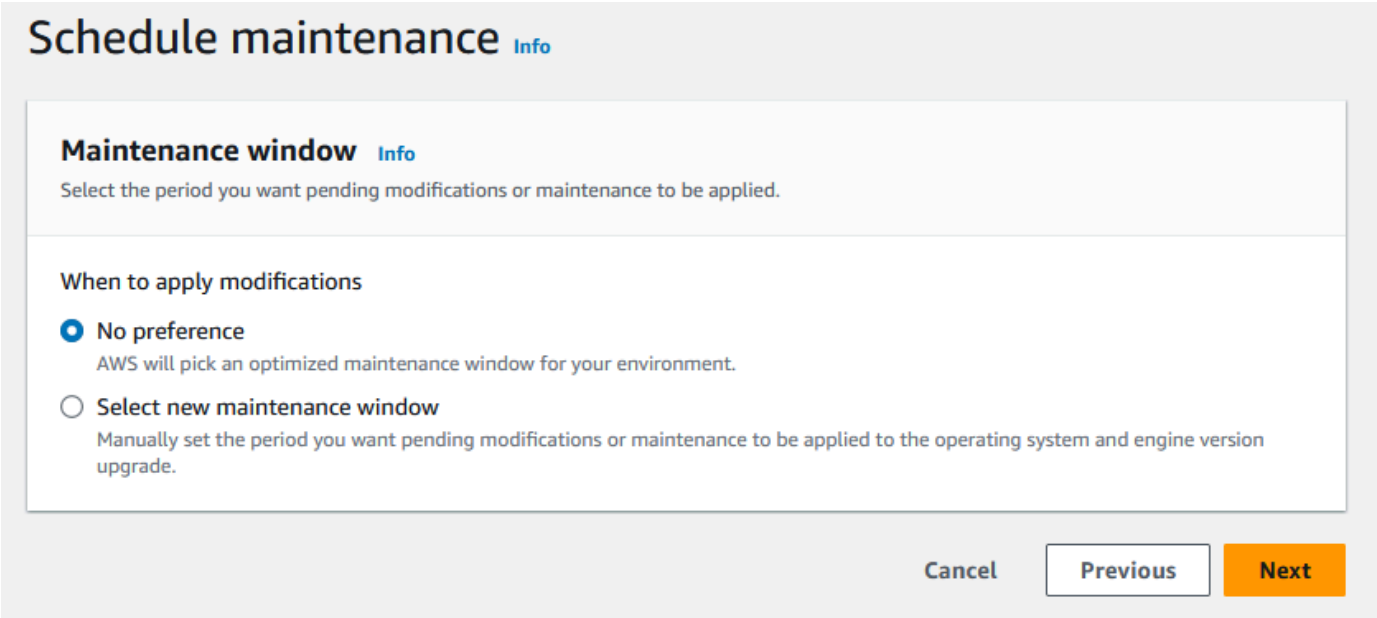
subnet-56f1a | us-west-2a

subnet-6685 | us-west-2b

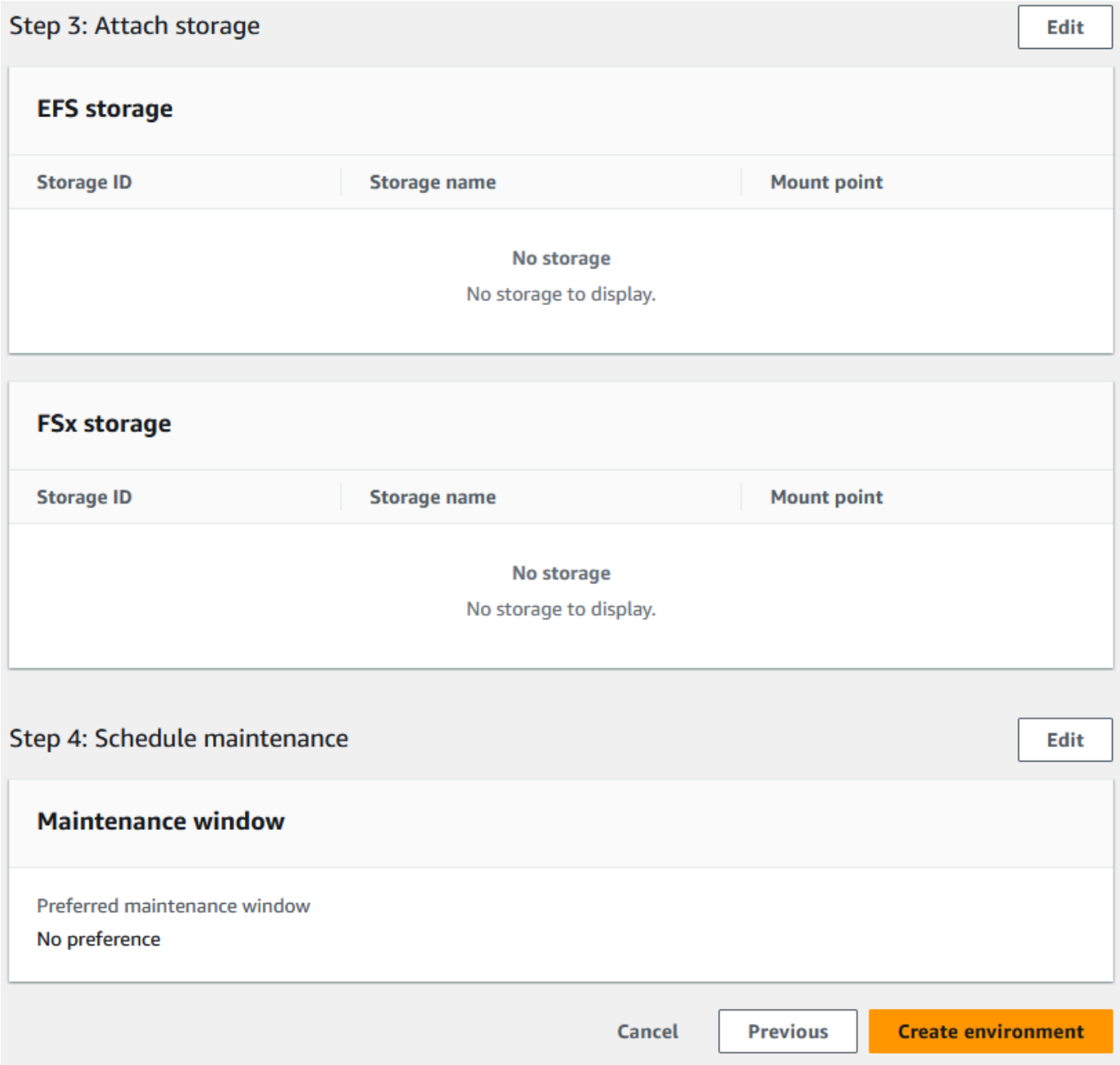
Security groups
Choose one or more security groups for the chosen VPC.

5. 在 [附加儲存空間] 頁面上，選擇 [下一步

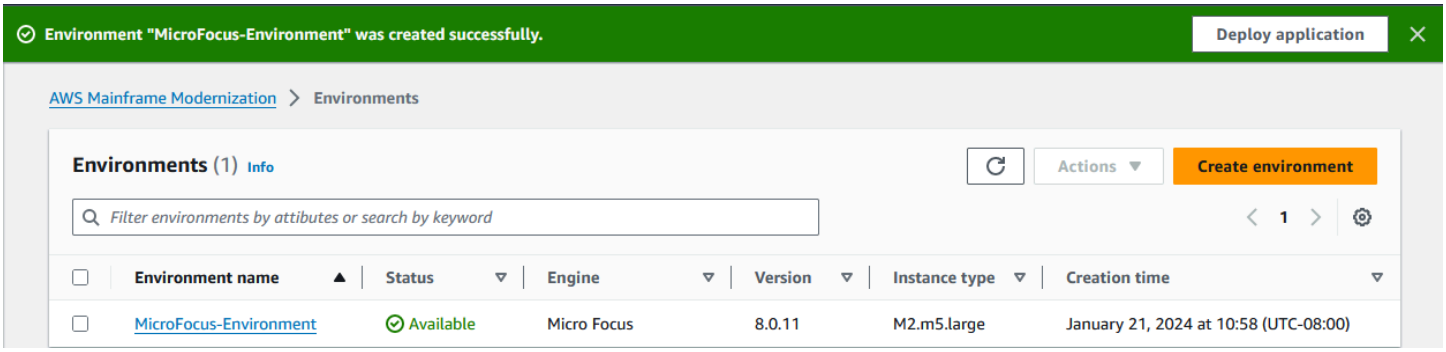
6. 在 [排程維護] 頁面上，選擇 [無偏好設定]，然後選擇 [下一步



7. 在 [檢閱並建立] 頁面上，複查您為程式實際執行環境提供的所有組態，然後選擇 [建立環境]。



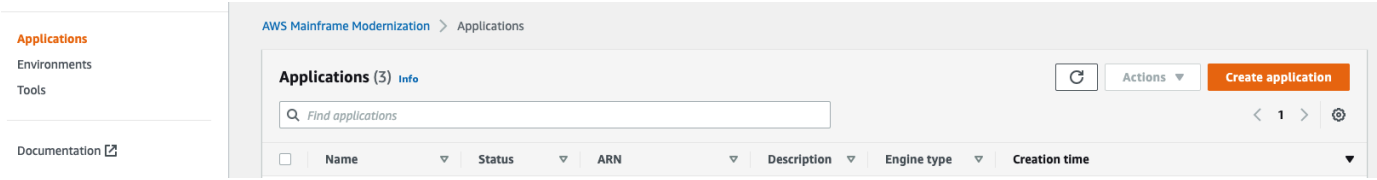
當您建立環境之後，會出現一個橫幅`Environment` *`name`* `was created successfully`，並且 [狀態] 欄位會變更為 [可用]。環境建立程序需要幾分鐘，但您可以在執行時繼續執行後續步驟。



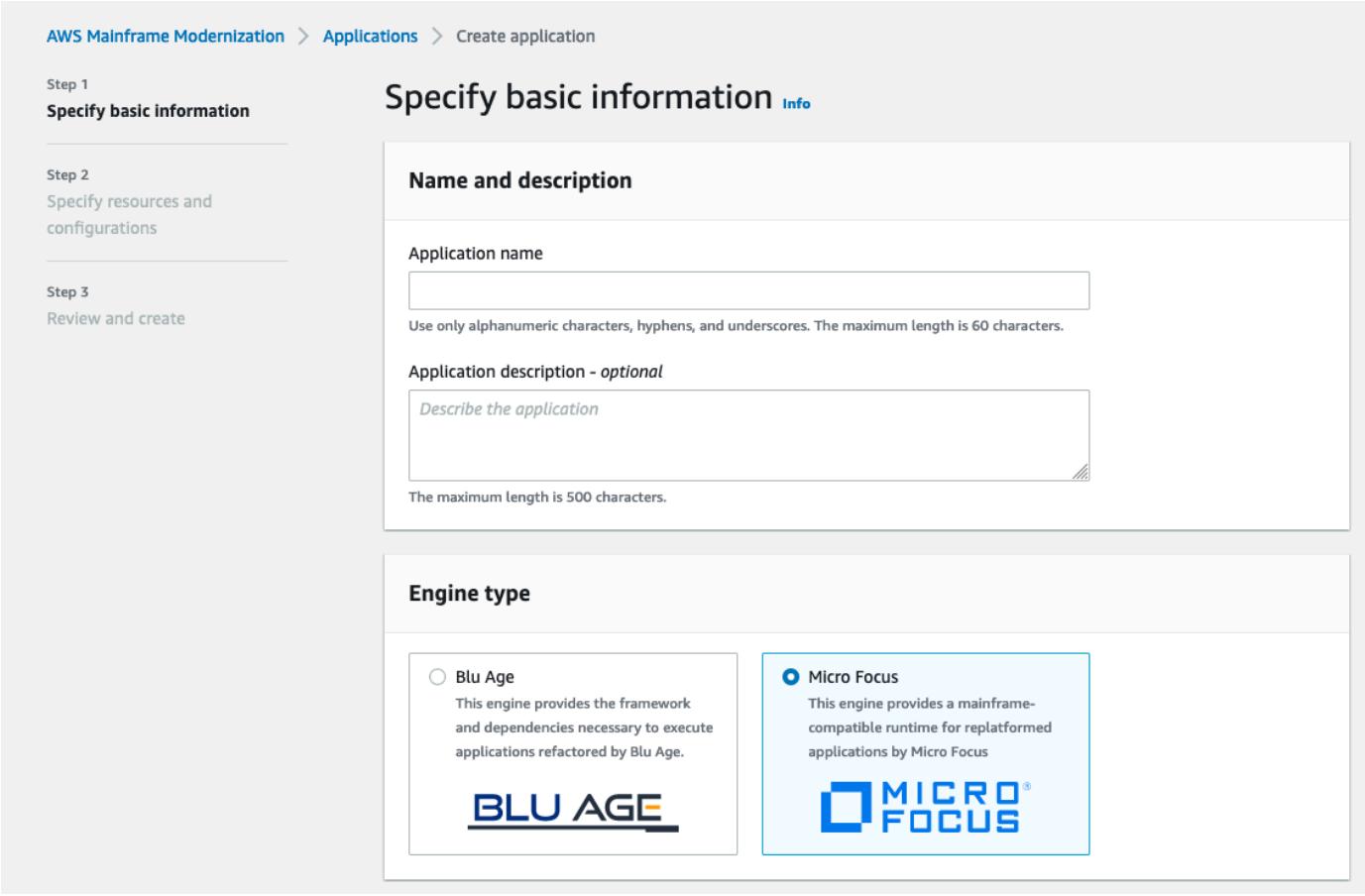
步驟 6：建立應用程式

建立應用程式

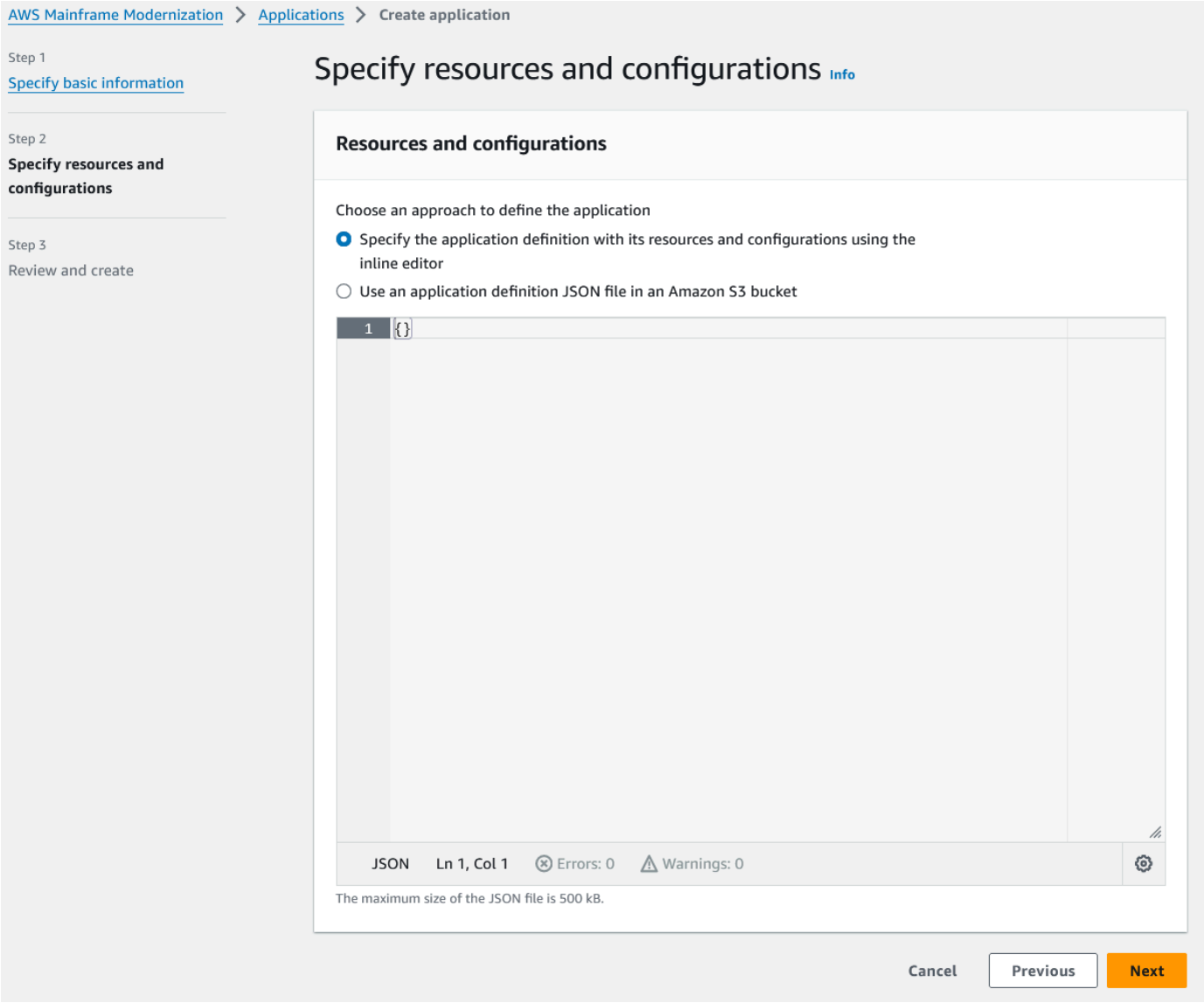
1. 在導覽窗格中，選擇 Applications (應用程式)。然後選擇創建應用程序。



2. 在 [建立應用程式] 頁面的 [指定基本資訊] 下，輸入`MicroFocus-CardDemo`應用程式名稱，並在 [引擎類型] 下確定已選取 Micro Focus。然後選擇下一步。



3. 在 [指定資源和組態] 下，選擇使用內嵌編輯器指定應用程式定義及其資源和組態的選項。



在編輯器中輸入下列應用程式定義：

```
{
  "template-version": "2.0",
  "source-locations": [
    {
      "source-id": "s3-source",
      "source-type": "s3",
      "properties": {
        "s3-bucket": "yourname-aws-region-carddemo",
        "s3-key-prefix": "CardDemo"
      }
    }
```

```
  ],
  "definition": {
    "listeners": [
      {
        "port": 6000,
        "type": "tn3270"
      }
    ],
    "dataset-location": {
      "db-locations": [
        {
          "name": "Database1",
          "secret-manager-arn":
 "arn:aws:secretsmanager:Region:123456789012:secret:MicroFocus-Tutorial-RDS-Secret-
xxxxxx"
        }
      ]
    },
    "batch-settings": {
      "initiators": [
        {
          "classes": [
               "A",
               "B"
            ],
          "description": "initiator_AB...."
        },
        {
          "classes": [
                "C",
                "D"
             ],
          "description": "initiator_CD...."
        }
      ],
      "jcl-file-location": "${s3-source}/catalog/jcl"
    },
    "cics-settings": {
      "binary-file-location": "${s3-source}/loadlib",
      "csd-file-location": "${s3-source}/rdef",
      "system-initialization-table": "CARDSIT"
    },
    "xa-resources": [
      {
```

```
        "name": "XASQL",
        "secret-manager-arn":
 "arn:aws:secretsmanager:Region:123456789012:secret:MicroFocus-Tutorial-RDS-Secret-
xxxxxx",
        "module": "${s3-source}/xa/ESPGSQLXA64.so"
      }
    ]
  }
}
```

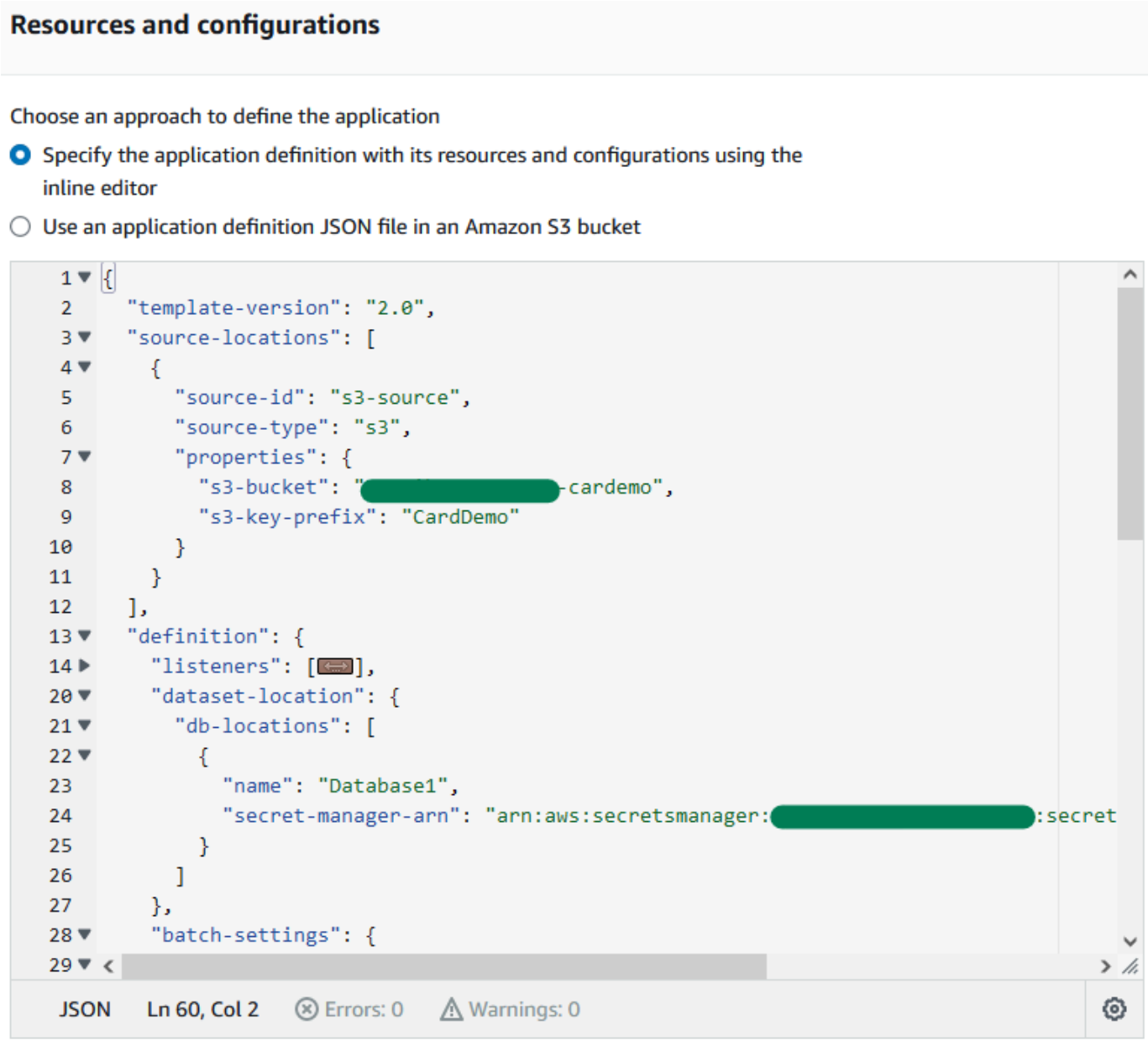
> **Note**
>
> 此檔案可能會變更。

4. 在源位置的屬性對象JSON中編輯應用程序，如下所示：

   a. 將的值取代為`s3_bucket`您在步驟 1 中建立的 Amazon S3 儲存貯體的名稱。

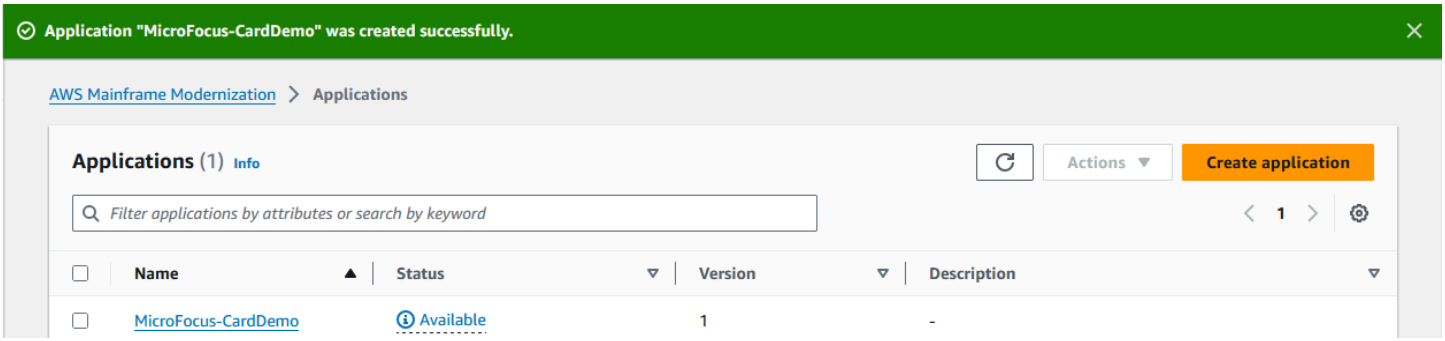
   b. 將的值取代為`s3-key-prefix`您上傳 CardDemo 範例檔案的資料夾 (key prefix)。如果您將目`CardDemo`錄直接上傳到 Amazon S3 儲存貯體，則`s3-key-prefix`不需要變更。

   c. 將這兩個`secret-manager-arn`值取代ARN為您在步驟 4 中建立的資料庫密碼。



如需應用程式定義的詳細資訊，請參閱微焦點應用定義。

5. 選擇 Next (下一步) 繼續。
6. 在 [檢閱並建立] 頁面上，檢閱您提供的資訊，然後選擇 [建立應用程式]。

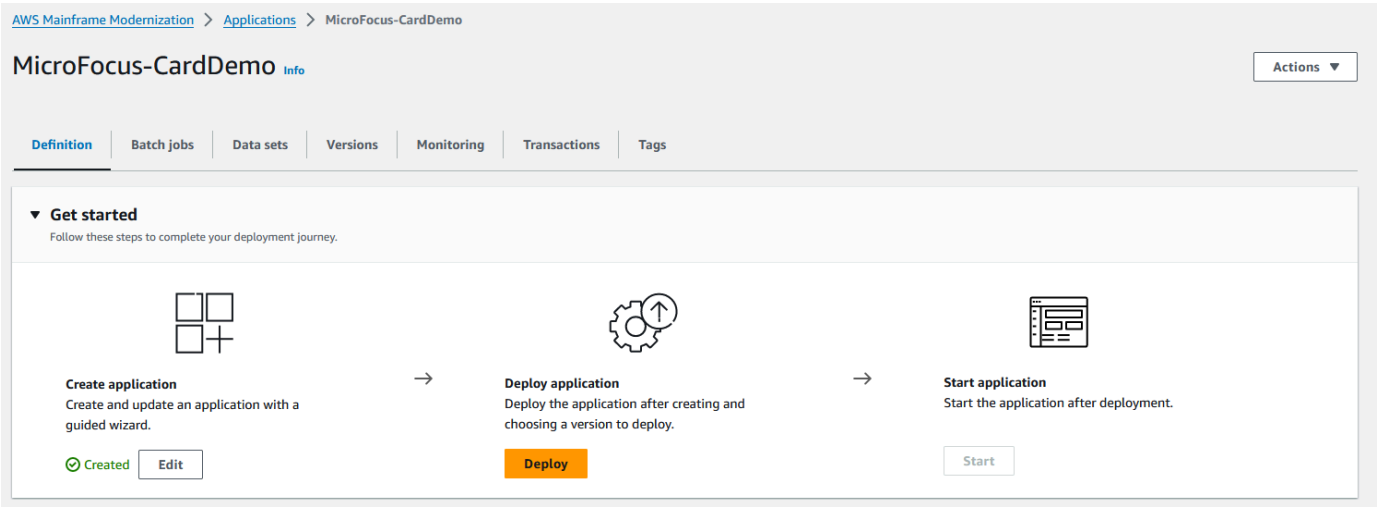


創建應用程序後，會出現一個橫幅，上面寫著Application *name* was created successfully。「狀態」字段將更改為「可用」。

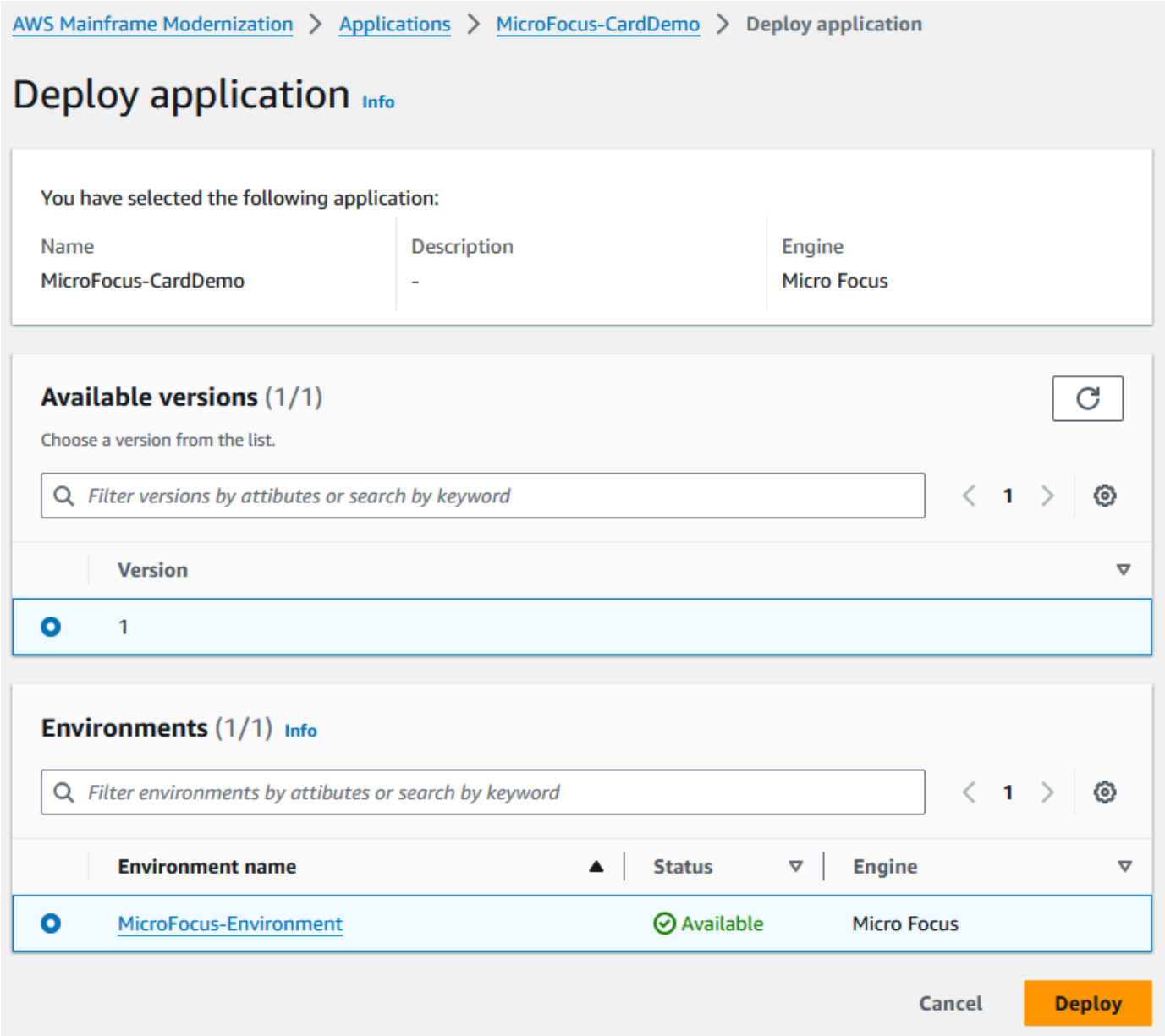
步驟 7：部署應用程式

若要部署應用程式

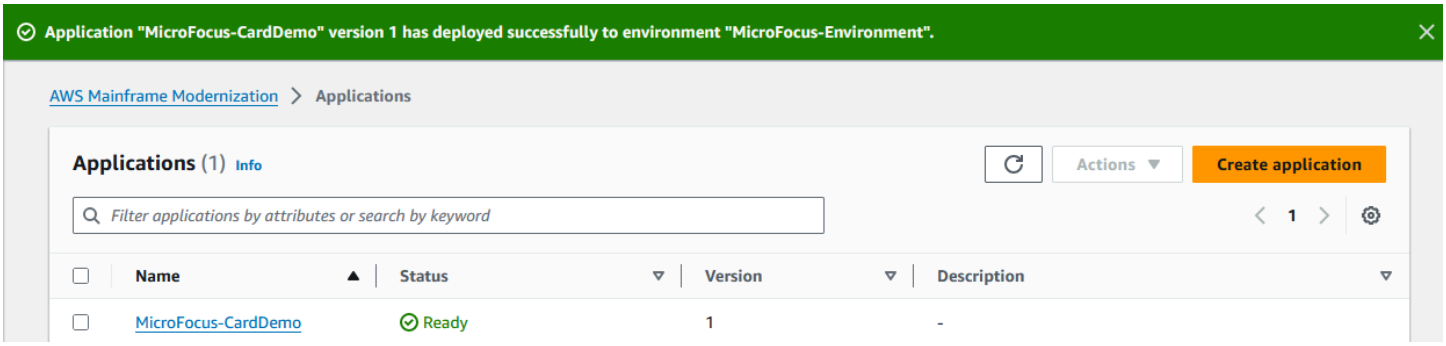
1. 在功能窗格中，選擇 [應用程式]，然後選擇MicroFocus-CardDemo。
2. 在部署應用程式下，選擇部署。



3. 選擇最新版本的應用程式和您先前建立的環境，然後選擇 [部署]。



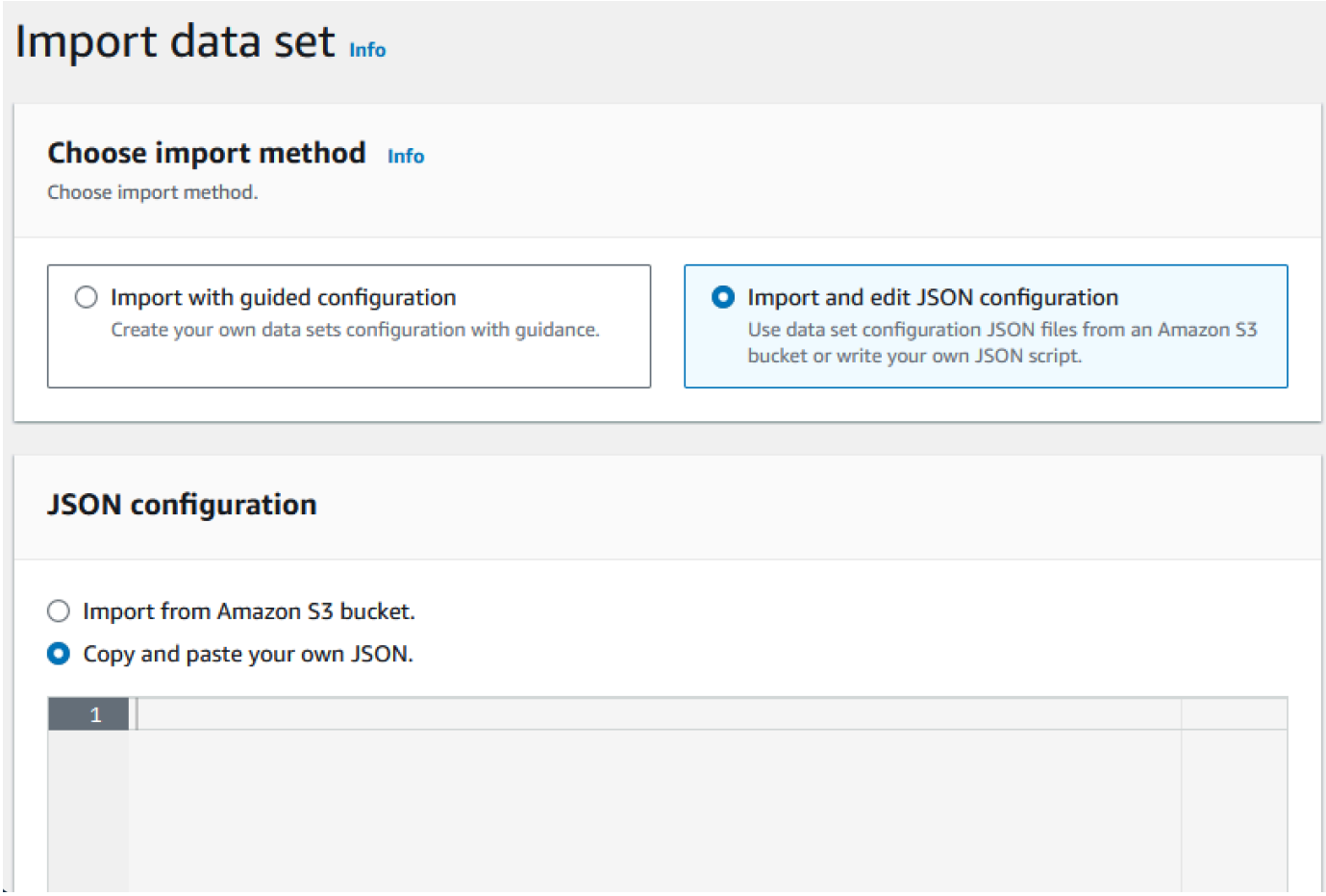
成功部署 CardDemo 應用程式時，狀態會變更為 [就緒]。



步驟 8：匯入資料集

匯入資料集的步驟

1. 在功能窗格中，選擇 [應用程式]，然後選擇應用程式。
2. 選擇「資料集」頁標。然後選擇 Import (匯入)。
3. 選擇 [匯入和編輯] JSON 設定，然後選擇 [複製並貼上您自己的JSON選項。



4. 複製並粘貼以下內容，JSON但不要選擇「提交」。這JSON包含示範應用程式所需的所有資料集，但需要 Amazon S3 儲存貯體詳細資訊。

```
{
    "dataSets": [
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.ACCTDATA.VSAM.KSDS",
                "relativePath": "DATA",
                "datasetOrg": {
```

```
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 11,
                            "offset": 0
                        }
                    }
                },
                "recordLength": {
                    "min": 300,
                    "max": 300
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo/catalog/data/
AWS.M2.CARDDEMO.ACCTDATA.VSAM.KSDS.DAT"
            }
        },
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.CARDDATA.VSAM.AIX.PATH",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 11,
                            "offset": 16
                        }
                    }
                },
                "recordLength": {
                    "min": 150,
                    "max": 150
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo/catalog/data/
AWS.M2.CARDDEMO.CARDDATA.VSAM.KSDS.DAT"
            }
        },
```

```
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.CARDDATA.VSAM.KSDS",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 16,
                            "offset": 0
                        }
                    }
                },
                "recordLength": {
                    "min": 150,
                    "max": 150
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo/catalog/data/
AWS.M2.CARDDEMO.CARDDATA.VSAM.KSDS.DAT"
            }
        },
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.CARDXREF.VSAM.KSDS",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 16,
                            "offset": 0
                        }
                    }
                },
                "recordLength": {
                    "min": 50,
                    "max": 50
                }
```

```
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo/catalog/data/
AWS.M2.CARDDEMO.CARDXREF.VSAM.KSDS.DAT"
            }
        },
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.CUSTDATA.VSAM.KSDS",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 9,
                            "offset": 0
                        }
                    }
                },
                "recordLength": {
                    "min": 500,
                    "max": 500
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo/catalog/data/
AWS.M2.CARDDEMO.CUSTDATA.VSAM.KSDS.DAT"
            }
        },
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.CARDXREF.VSAM.AIX.PATH",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 11,
                            "offset": 25
                        }
```

```
                    }
                },
                "recordLength": {
                    "min": 50,
                    "max": 50
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo/catalog/data/
AWS.M2.CARDDEMO.CARDXREF.VSAM.KSDS.DAT"
            }
        },
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 16,
                            "offset": 0
                        }
                    }
                },
                "recordLength": {
                    "min": 350,
                    "max": 350
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo/catalog/data/
AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS.DAT"
            }
        },
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.USRSEC.VSAM.KSDS",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
```

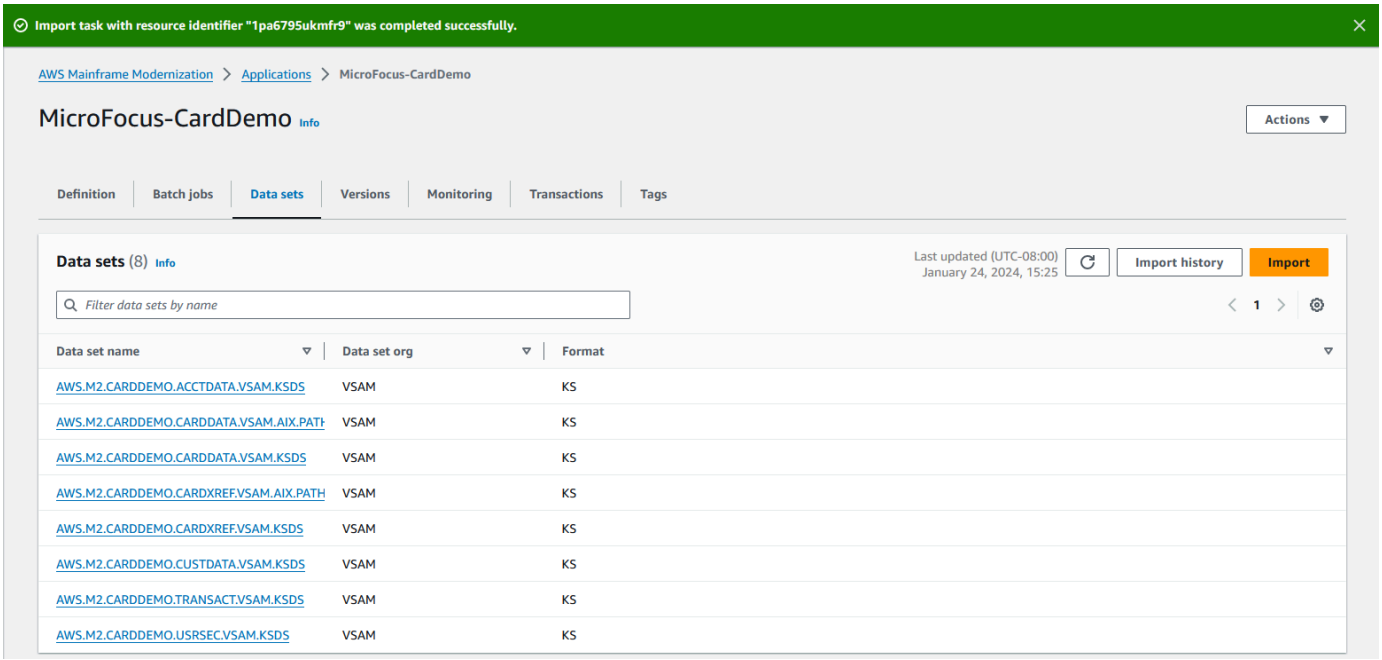
```
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 8,
                            "offset": 0
                        }
                    }
                },
                "recordLength": {
                    "min": 80,
                    "max": 80
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo/catalog/data/
AWS.M2.CARDDEMO.USRSEC.VSAM.KSDS.DAT"
            }
        }
    ]
}
```

5. 以包含 CardDemo 資料夾的 Amazon S3 儲存貯體的名稱取代每次出現的 `<s3-bucket-name>` (共有 8 個) `your-name-aws-region-carddemo`。

> **Note**
>
> 若要複製 Amazon S3 URI 中資料夾的 Amazon S3，請選取該資料夾，然後選擇「複製 Amazon S3」URI。

6. 選擇提交。

匯入完成時，會出現一個橫幅，其中包含下列訊息：`Import task with resource identifier` *`name`* `was completed successfully.`會顯示已匯入資料集的清單。

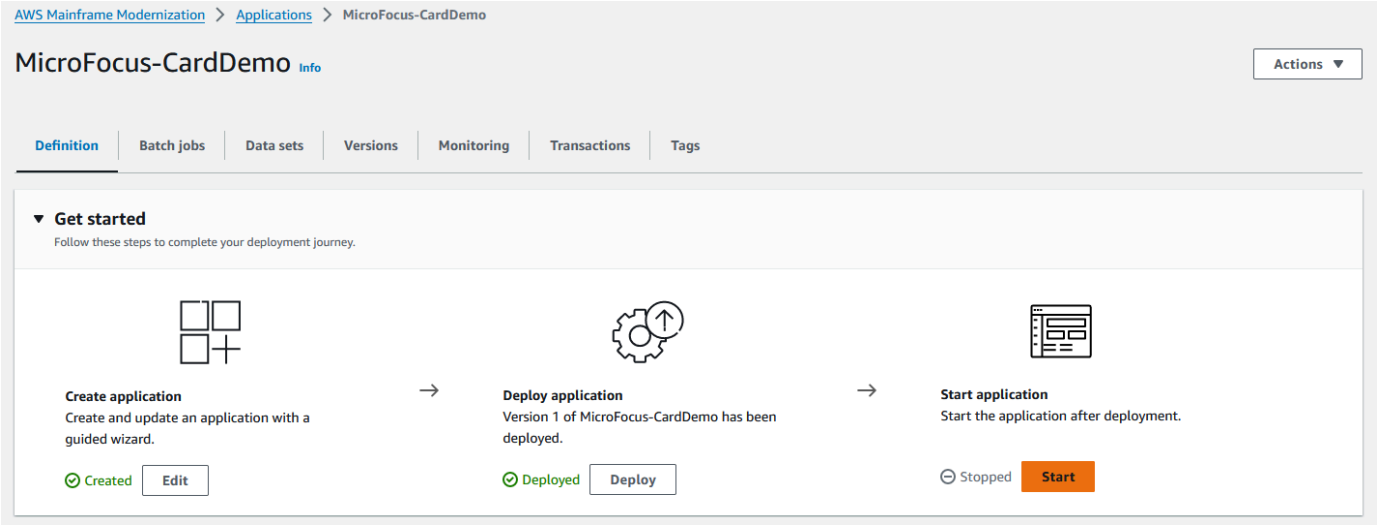


您也可以選擇「資料集」標籤上的「匯入歷史記錄」，來檢視所有資料集匯入的狀態。

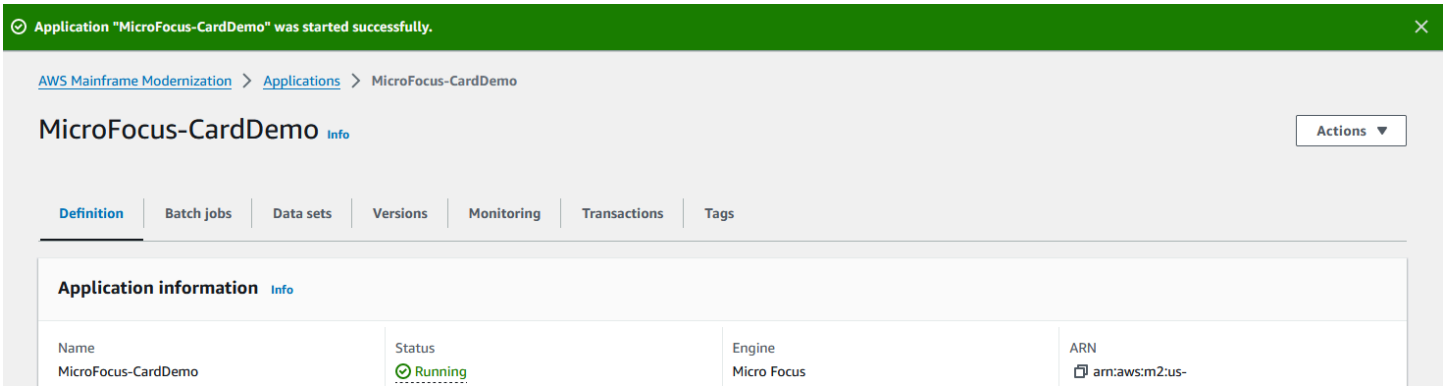
步驟 9：啟動應用程式

若要啟動應用程式

1. 在功能窗格中，選擇 [應用程式]，然後選擇應用程式。
2. 選擇啟動應用程式。



當 CardDemo 應用程式開始成功執行時，會出現一個橫幅，並顯示下列訊息：Application *name* was started successfully。「狀態」字段變為「正在運行」。



步驟 10：Connect 到 CardDemo CICS應用程序

在連線之前，請確定您為應用程式指定的VPC和安全性群組與您要連線的網路介面所套用的和安全性群組相同。

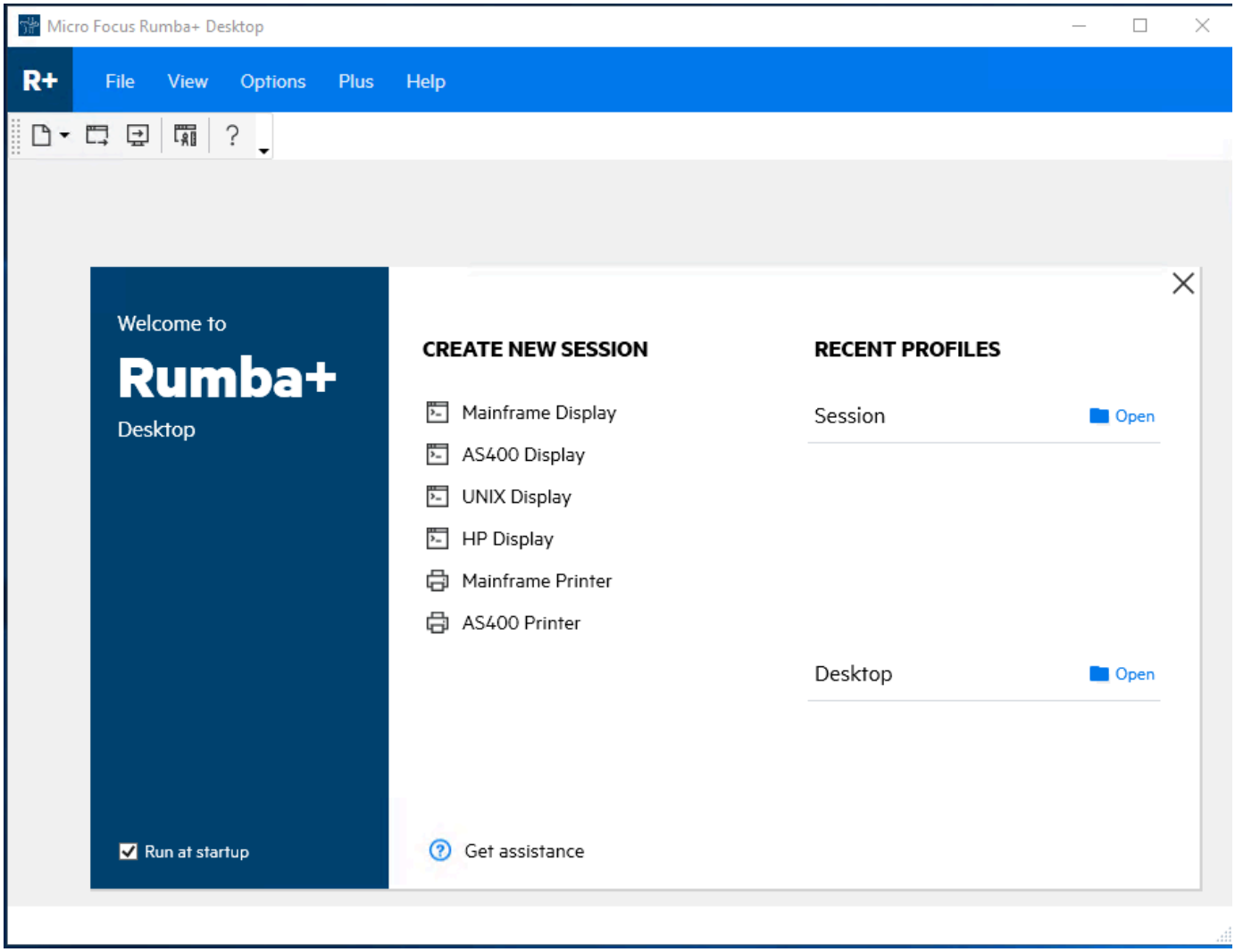
要配置 TN327 0 連接，您還需要應用程序的DNS主機名和端口。

使用終端機模擬器設定應用程式並將其連接至大型主機

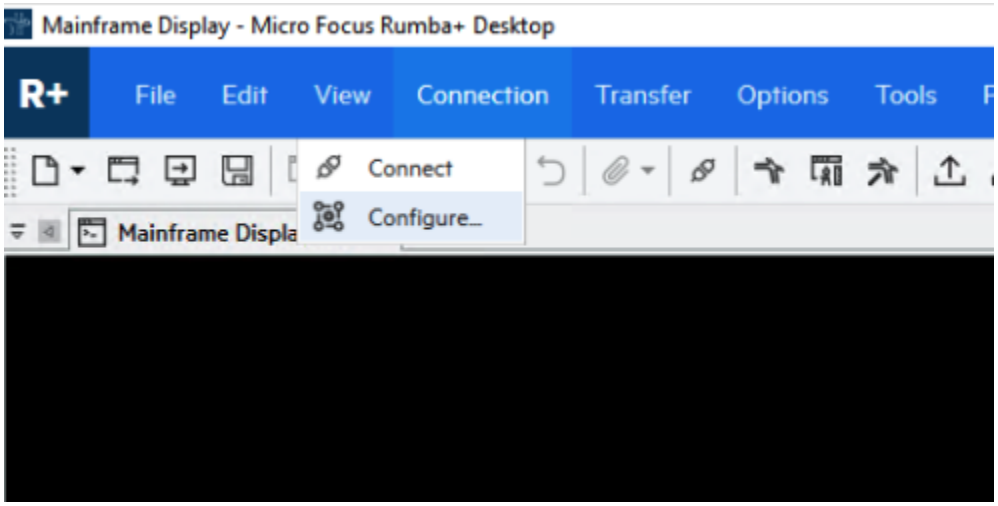
1. 開啟 AWS 大型主機現代化主控台並選擇 [應用程式]，然後選擇。MicroFocus-CardDemo
2. 選擇複製圖示以複製主DNS機名稱。另外，請確保記下端口號。
3. 啟動終端機模擬器。本教程使用微型焦點倫巴 +。

> **Note**
>
> 配置步驟因仿真器而異。

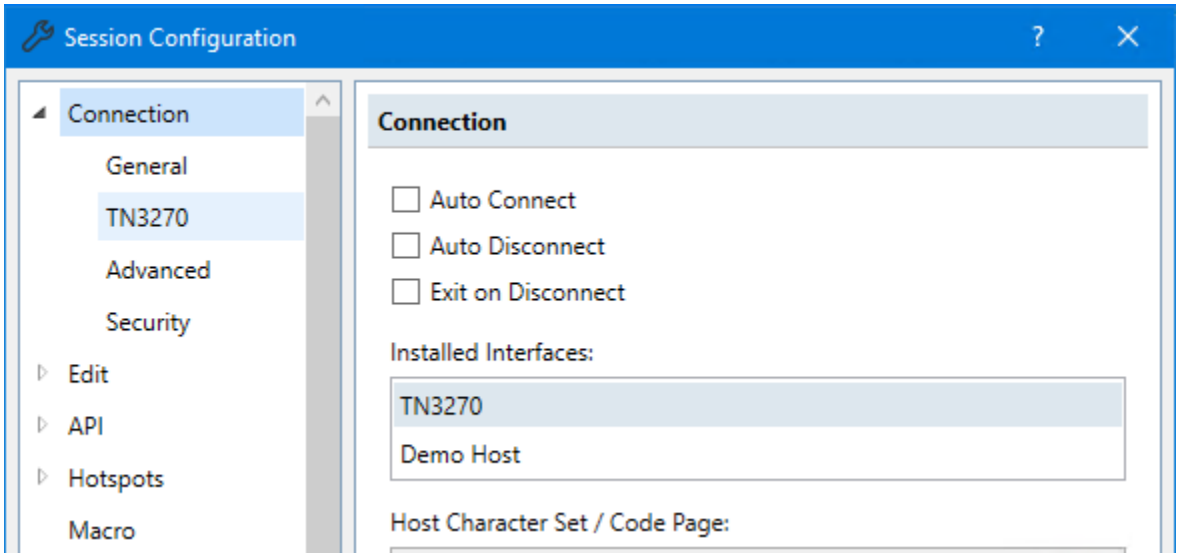
4. 選擇「大型主機顯示器」。



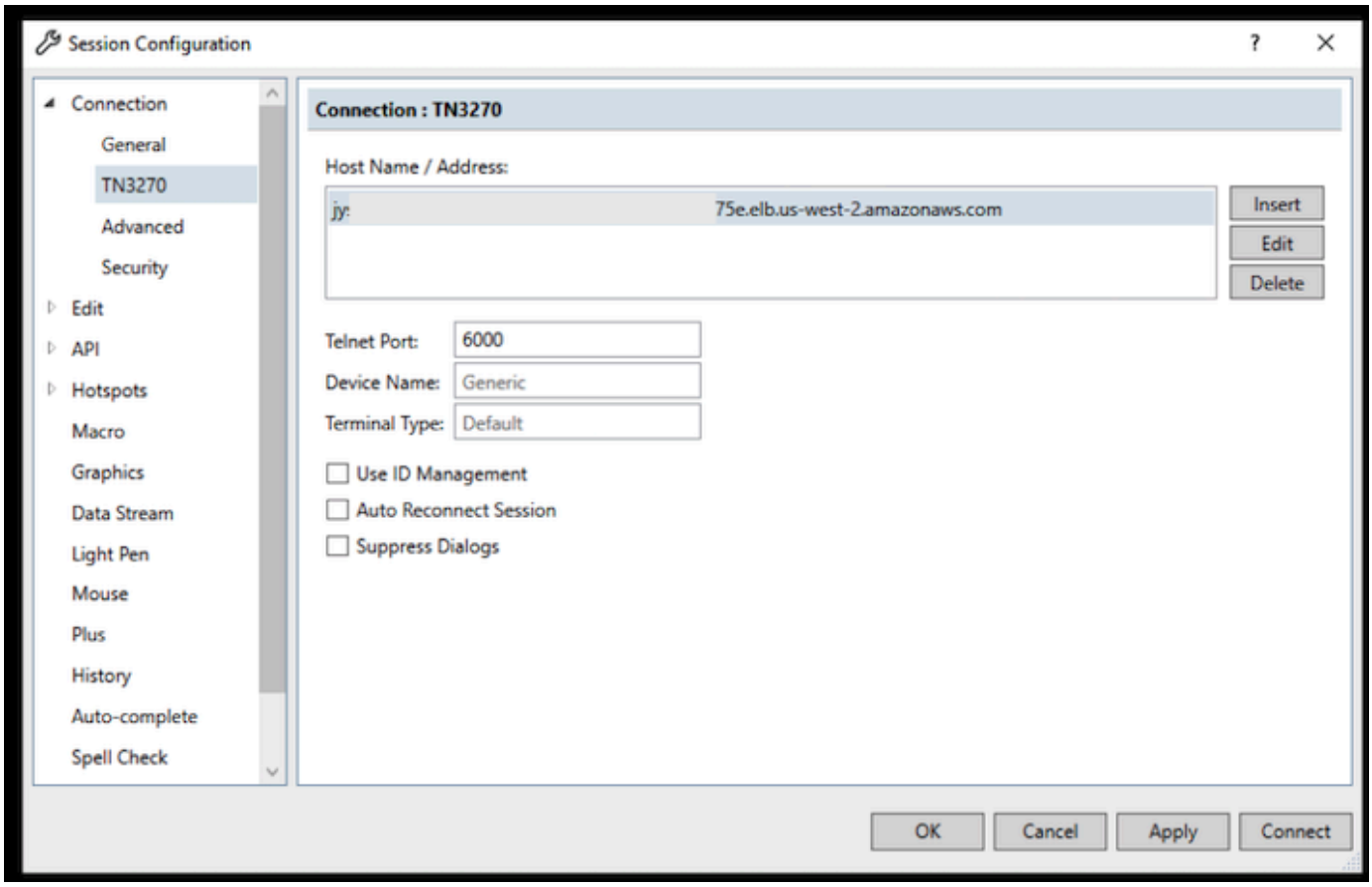
5. 選擇 [連線]，然後選擇 [設定]。



6. 在「已安裝的介面」下TN3270，選擇，然後在「連線」功能表下TN3270再次選擇。



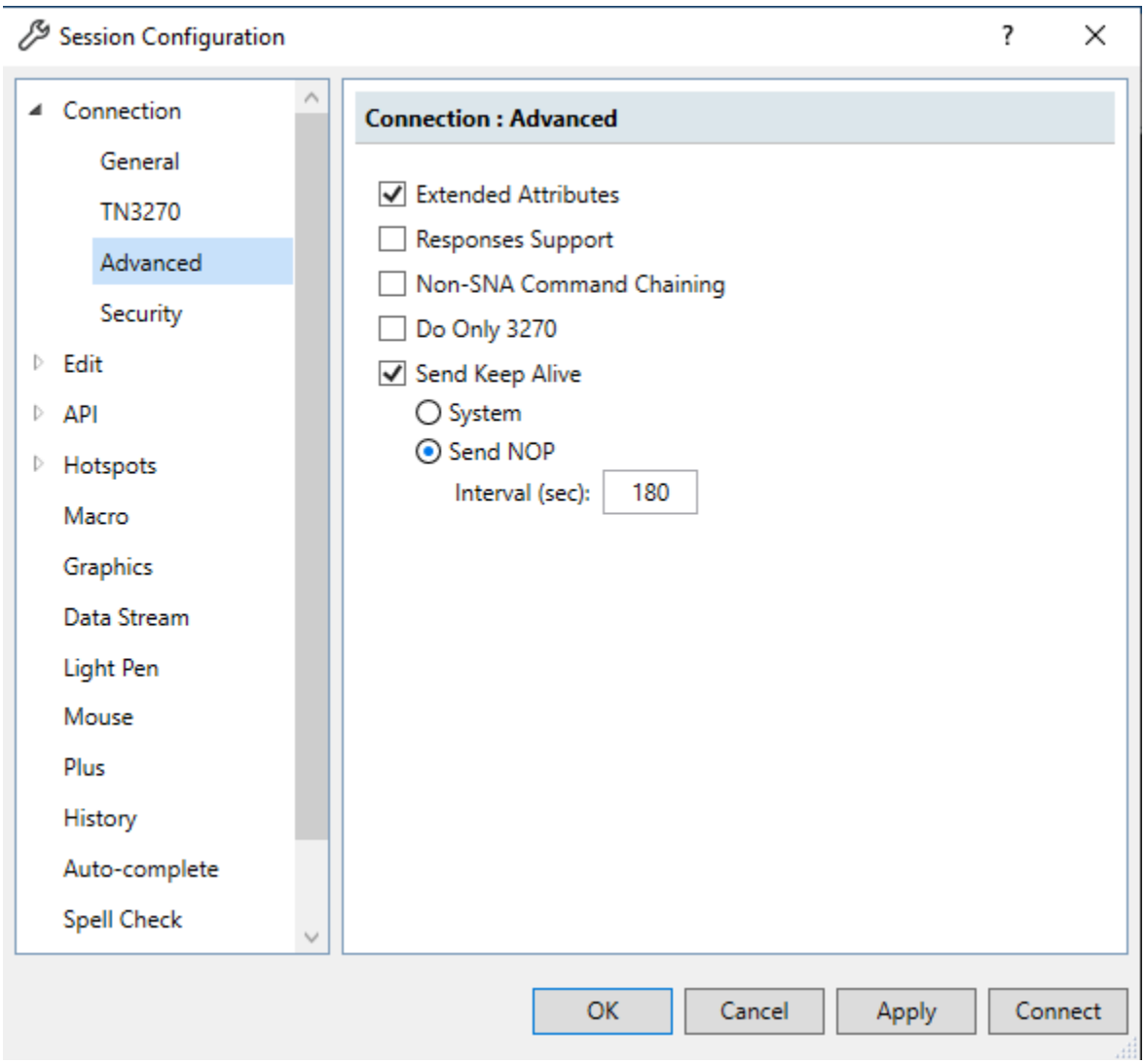
7. 選擇插入，然後粘貼應DNS Hostname用程序。指定 6000「Telnet 連接埠」的連接埠。



> **Note**
>
> 如果您在瀏覽器中使用 AWS AppStream 2.0，且在貼上值時遇到困難，請參閱疑難排解 AppStream 2.0 使用者問題。

8. 在 [連線] 底下，選擇 [進階]，然後選擇 [繼續傳送並傳送]NOP，然後輸入 180 做為 [間隔]。

> **Note**
>
> 將 TN327 0 終端機上的保持作用中設定至少 180 秒，有助於確保 Network Load Balancer 不會中斷您的連線。

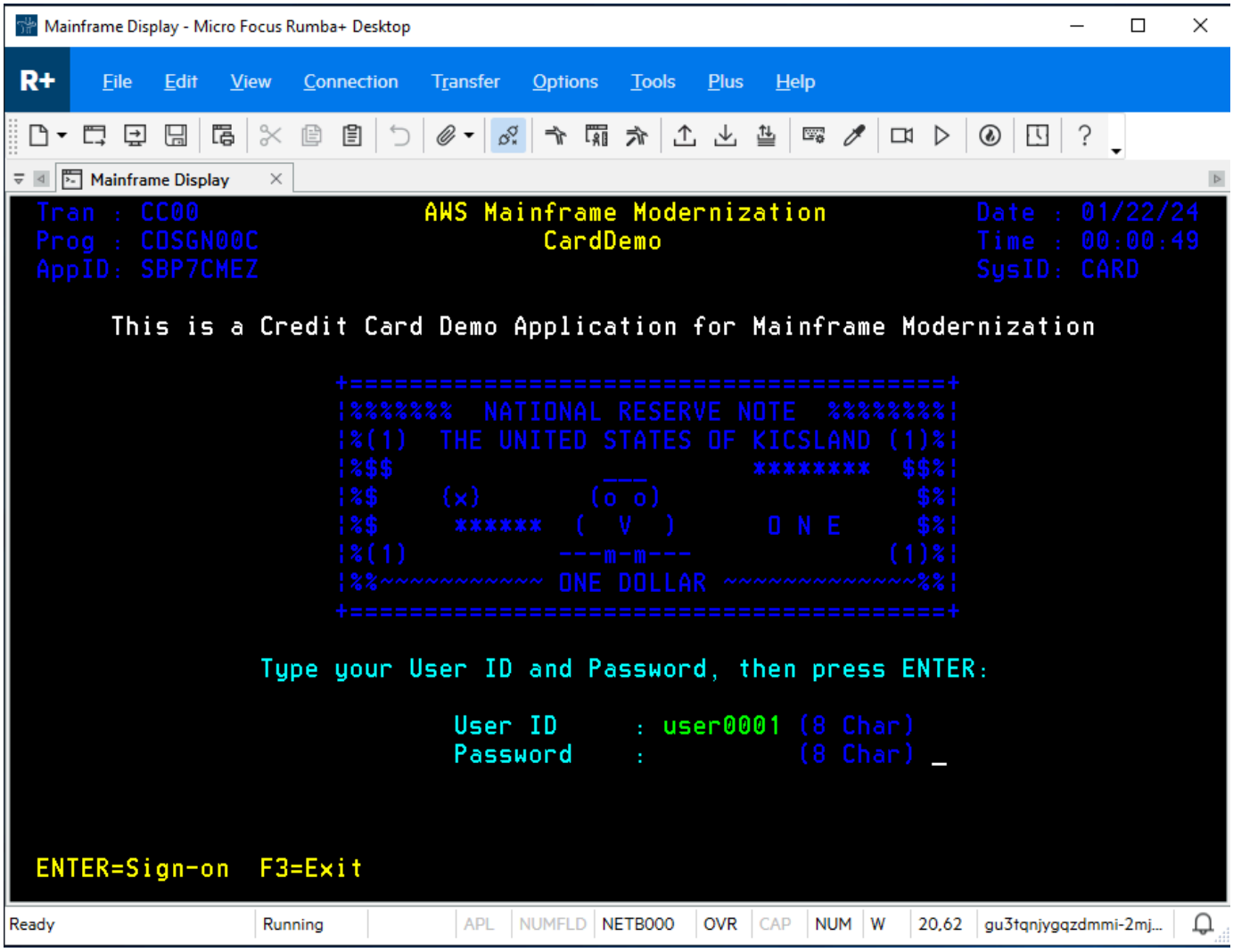


9. 選擇連線。

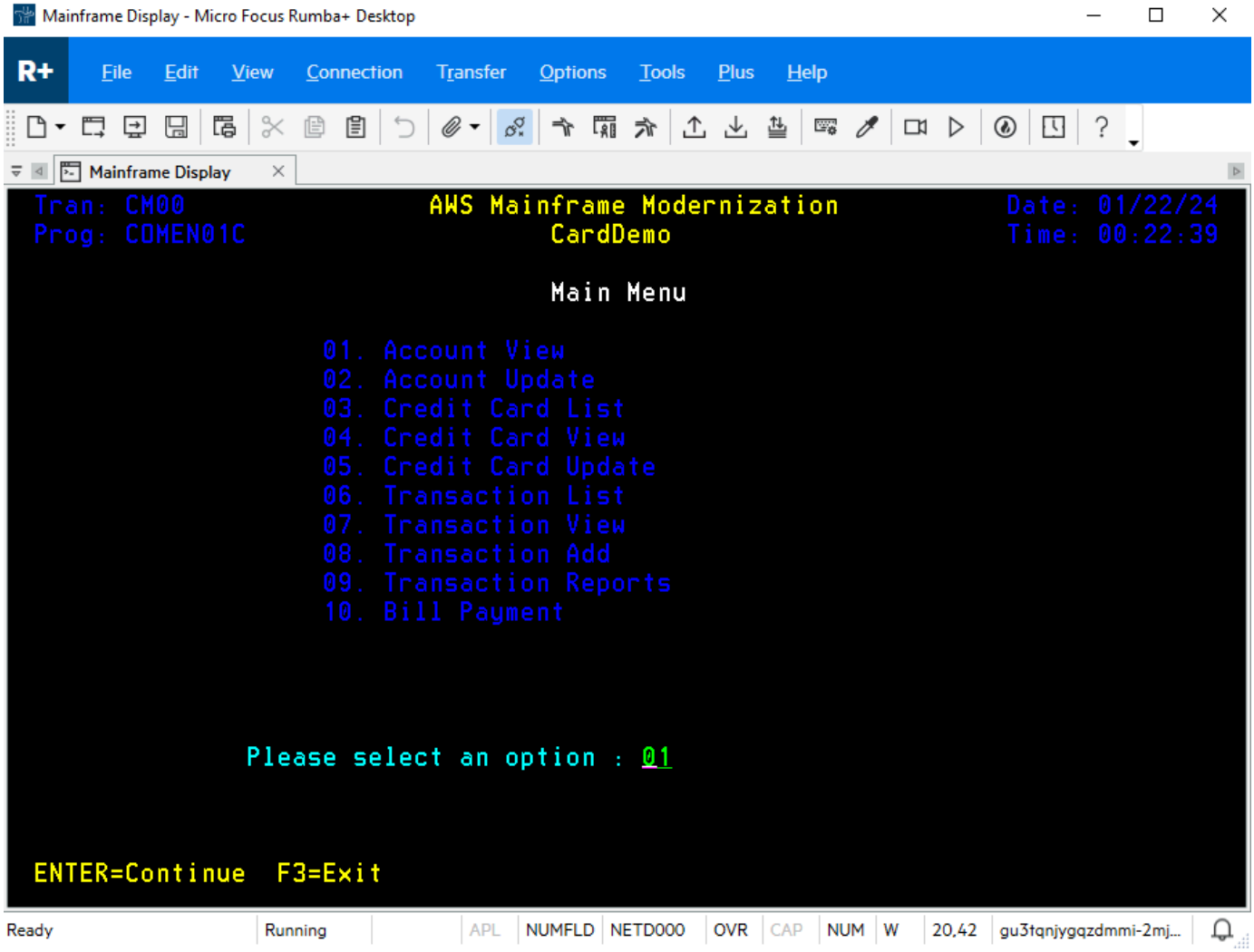
> **Note**
>
> 如果連線失敗：
>
> - 如果您使用 AppStream 2.0，請確認專為應用程式環境設定的VPC和安全性群組與 AppStream 2.0 叢集相同。
> - 使用 VPC Reachability Analyzer 分析連接。您可以透過主控台存取「可 Reachability Analyzer」。
> - 做為診斷步驟，請嘗試新增或變更應用程式的「安全群組」輸入規則，以允許來自任何位置的連接埠 6000 流量 (亦即CIDR封鎖 0.0.0/0)。如果您成功連線，您就會知道安全性群組封鎖了您的流量。將安全性群組來源變更為更具體的內容。如需有關安全性群組的詳細資訊，請參閱安全性群組基礎

10. 輸入使USER0001用者名稱和password密碼。

> **Note**
>
> 在倫巴語中，「清除」的預設值為 ctrl-shift-z，「重設」的預設值為 ctrl-r。

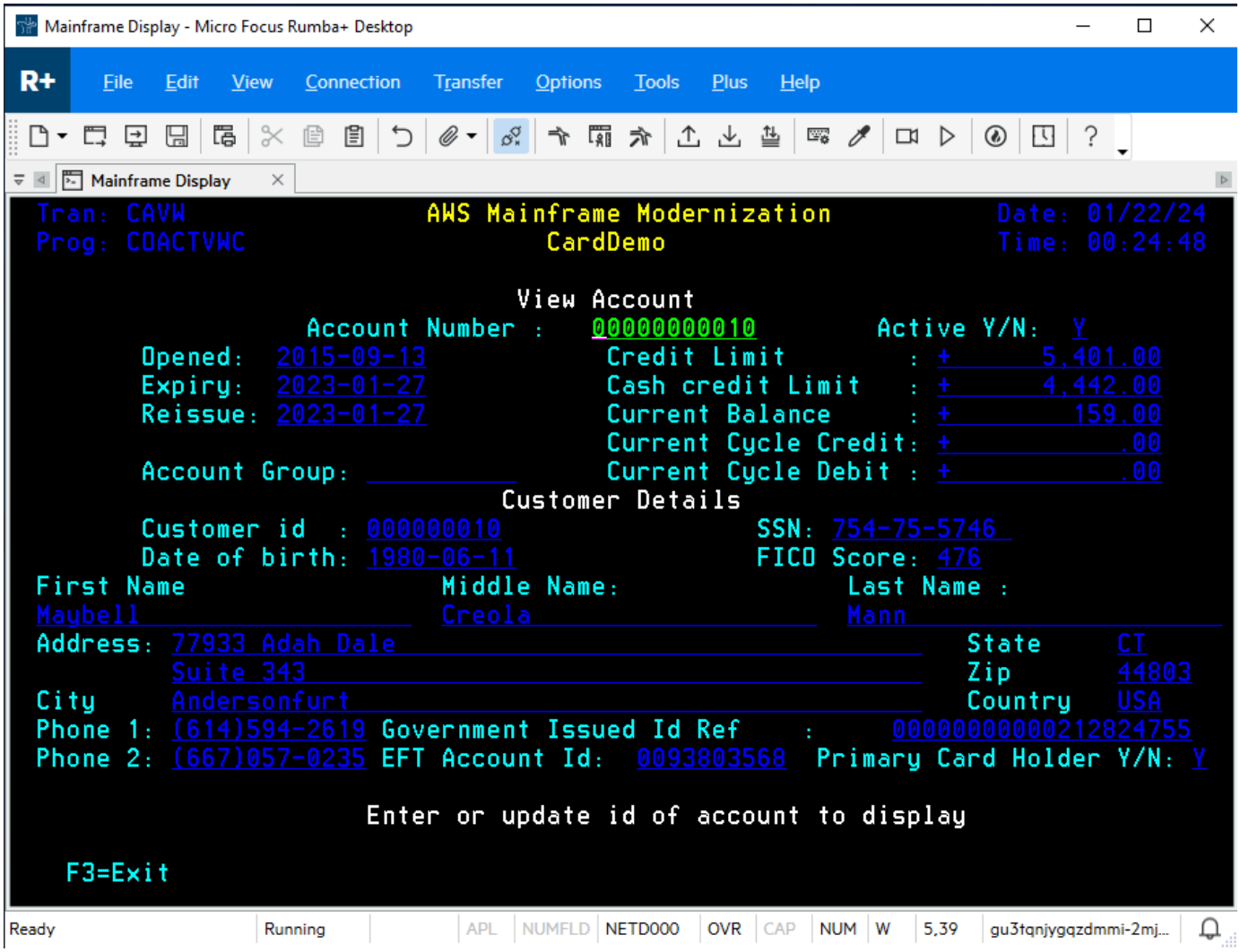


11. 成功登入後，您可以瀏覽 CardDemo應用程式。
12. 01針對「帳戶檢視」輸入。



13. 輸入00000000010帳戶號碼，然後按鍵盤上的 Enter 鍵。

> **Note**
>
> 其他有效帳戶為0000000011和00000000020。



14. 按退出F3到菜單，然F3後退出交易。

清除資源

如果您不再需要為此教學課程建立的資源，請刪除這些資源以避免額外費用。若要這樣做，請完成下列步驟：

- 如有必要，請停止應用程式。
- 刪除 應用程式。如需詳細資訊，請參閱刪除 AWS Mainframe Modernization 應用程式。
- 刪除執行階段環境。如需詳細資訊，請參閱刪除 AWS 大型主機現代化執行階段環境。
- 刪除您為本教學建立的 Amazon S3 儲存貯體。如需詳細資訊，請參閱 Amazon S3 使用者指南中的刪除儲存貯體。

- 刪除您為此教學課程建立的 AWS Secrets Manager 密碼。如需詳細資訊，請參閱刪除密碼。
- 刪除您為此教學課程建立的KMS金鑰。如需詳細資訊，請參閱刪除AWSKMS金鑰。
- 刪除您為本教學課程建立的 Amazon RDS 資料庫。EC2如需詳細資訊，請參閱 Amazon RDS 使用者指南中的刪除執行個體和資料庫執行個體。
- 如果您新增連接埠 6000 的安全性群組規則，請刪除該規則。

後續步驟

若要瞭解如何為您的現代化應用程式設定開發環境，請參閱教學課程：設定 AppStream 2.0 以搭配 Micro Focus 企業分析器和 Micro Focus 企業開發人員使用。

使 AWS 用藍光時代自動重構應用程序

使用 AWS Blu Age 進行自動化重構提供移轉和現代化大型主機應用程式的 end-to-end 解決方案。在重構過程中的步驟如下：

- 分析庫存
- 分析相依性
- 自動轉換程式碼
- 擷取和管理測試案例

您可以在 Blu Insights 工具中完成前面的步驟，該工具可透過單一登入從大型主機現代化 AWS 主控台取得。有關藍光洞察的更多信息，請參閱 Blu 洞察文檔。

當您對轉換後的源代碼感到滿意時，是時候移動到了 AWS，您將在其中完成以下步驟：

- 建置和部署重構的應用程式。
- 在 AWS 大型主機現代化中部署和監控您的應用程式。

AWS 藍光時代運行時（非託管）是大 AWS 型主機現代化服務以及 Blu Age 管理的產品之一 AWS 。透過 AWS Blu Age Managed，您可以將現代化應用程式部署到可簡化體驗的 AWS管理環境中，因此您不需要管理執行現代化應用程式的基礎架構。相反，使用 AWS Blu Age Runtime（非託管），您可以在自己的 AWS 帳戶中部署現代化的應用程序，以便管理自己的基礎架構。使用 AWS Blu Age Runtime（非託管），您可以靈活地以自己的方式運行現代化應用程序所需的所有技術組件。

AWS 藍光年齡運行時（非託管）可在以下位置部署：

- Amazon EC2
- Amazon ECS 在 Amazon EC2
- Amazon EKS 在 Amazon EC2
- Amazon ECS 管理 AWS Fargate

在 Amazon 上部署 EC2 (上述清單中的前三個選項) 可以直接在執行個體中或透過 Docker 容器化應用程式完成，這是使用 Amazon 或 Amazon ECS 時的首選方式。EKS

主題

- AWS 藍光時代版本資訊
- AWS 藍光時代的升級說明
- AWS 藍色時代運行時概念
- 為 AWS 藍光時代運行時設置配置
- AWS 藍色時代運行時 APIs
- 設置 AWS 藍光時代運行時（非託管）
- 使用藍光時代開發人員修改源代碼 IDE

AWS 藍光時代版本資訊

本節包含 AWS Blu Age 執行階段和現代化工具的發行說明，從 3.5.0 版開始，最新版本優先，按版本編號組織。

> **Note**
>
> 有關本文檔之前的版本說明，請聯繫 AWS Blu Age 交付服務。有關最新藍光洞察功能的信息，請參閱 Blu 洞察發布。

主題

- 發行版本公告 4.2.0
- 執行階段版本 4.2.0
- 現代化工具 4.2.0 版
- 發行版本公告 4.1.0
- 執行階段版本 4.1.0
- 現代化工具 4.1.0 版
- 發行版本公告 4.0.0
- 執行階段版本 4.0.0
- 現代化工具 4.0.0 版
- 發行版本公告
- 執行階段版本 3.10.0
- 現代化工具版本 3.10.0

- 發行版本公告
- 執行階段版本 3.9.0
- 現代化工具 3.9.0 版
- 發行版本公告
- 執行階段版本 3.8.0
- 現代化工具 3.8.0 版
- 發行版本公告
- 執行階段版本 3.7.0
- 現代化工具 3.7.0 版
- 發行版本公告 3.6.0
- 執行階段版本 3.6.0
- 現代化工具 3.6.0 版
- 發行版本公告 3.5.0
- 執行階段版本 3.5.0
- 現代化工具 3.5.0 版

發行版本公告 4.2.0

發行日期：二零二四年七月十日

此版本的 AWS Blu Age 運行時和現代化工具專注於性能和安全性。此版本中的一些主要功能和變更包括：

- 我們改善了轉換效能，特別是針對具有超過 3,000 萬行程式碼的大型專案。我們實施了一系列改進，我們獲得的結果顯示時間減少了 150% 以上，並在幾分鐘內完成運行，而不是幾小時。我們實施的關鍵改進是配置超時機制，以限制分配給分析的時間上限，從而跳過檢測到問題的文件。我們標記跳過的文件，以便您可以在必要時稍後調查它們。
- 我們增加了對 AS4 00 個項目的分佈式鎖定管理系統的支持。在應用程式的多個執行個體以相同資料庫為目標的高可用性環境 (多節點) 中，在這些執行個體的整個生命週期中維持資料一致性是一項重大挑戰。為了有效解決這項挑戰，我們將 Redis 新增為共用和外部快取伺服器，以便在批次處理模式下執行時在所有執行個體之間進行協調。
- 我們為表格元件增加了一個新的動態分頁功能。此功能的目標是縮短回應時間，並減少具有大量資料列之資料表的記憶體使用量。此功能可讓資料表元件只載入部分資料，並在瀏覽頁面時隨選擷取更多

記錄。為了進一步改善體驗，該平台還支持數據的預取。這項新的動態分頁功能為具有大型資料集的應用程式提供更有效率且回應更快的使用者體驗。

- 為了解決經常出現的關鍵挑戰，我們增加了對嵌套COBOL程序的支持。以前，現代化嵌套COBOL程序的解決方法涉及手動將程序分離為不同的文件，通過鏈接部分鏈接它們，並使它們使用必要的參數相互調用。這個過程不僅耗時，而且容易出錯。您現在可以將巢狀COBOL程式現代化，而不需要手動分離。

我們使用以下堆棧測試了此版本的 AWS Blu Age 運行時。其他元件版本也可能相容。

元件	版本
Java	Java 17
表示層	節點 JS 18.18, 故宮 9.8, 角 17
服務層	彈簧靴子 3.2.4, 彈簧核心 6.1.5, 彈簧啟動器 4.0.0
持久層	波斯特雷SQL引擎 14
應用程式伺服器	阿帕奇雄貓 10.1.17

如需有關此發行版本中所含變更的詳細資訊，請參閱下列各節。

執行階段版本 4.2.0

ZoS

新功能

- DB2-支援查詢中沒有結構描述限定詞的SQL預存程序叫用
- COBOL-增加了對 HEX-OF 功能的支持
- COBOL-增加了對嵌套程序的支持
- COBOL-增加了對 FUNCTION TEST-YYYYMMDD 和 DATE TEST--的支持 DAY YYYYDDD
- CICS-增加了對SETTERMINAL命令UCTRANST中的選項的支持
- CICS-增加了對INQUIREDB2CONN命令的支持

- BluSam -增加了對動態訪問鍵刪除的支持 VSAM
- IMS-增加了對TERM命令的支持
- BAC-在所有BACREST端點上新增授權檢查
- BAC-增加了配置，application-main.yaml以定義記錄大小以過濾符合該記錄大小的加載掩碼
- BAC和JAC：添加了配置，通過指定application-main.yaml來檢索密碼中默認超級管理員用戶的用戶名和密碼 ARN

改善

- JCL-SORT-增強了對條OMIT款的支持，以處理帶有 Shiftin 和字符的條件 ShiftOut
- JCL-SORT-改進了對BDW現場的支持
- JCL-SORT-改進了對與字段的多個GDG串聯的支持 BDW
- JCL-DFSORT-增INRECPARSESTARTAFT加了對/條款的STARTAT支持
- JCL-IEBGENER-增強了輸出文件的 recordSize 處理
- JCL-INFUTILB-禁用NULLINDICATOR基於 YML-FIX GRAPHIC CASE
- JCL-改善對於 FormatterParser 在OUTREC欄位中處理常數的支援
- JCL-增強了DSNUTILB程序實用程序中圖形類型的加載數據
- JCL-SORT-增強了對分區十進制格式的支持
- JCL-SORT-增強了對OMIT條款的支持，以處理帶有 Shiftin 和字符的條件 ShiftOut
- MQ-改進了 MQ 連接的處理以適應多個業務工作流程
- CICS-增強對 EXEC CICS READ SET (ptr-ref) 陳述式的指標參考支援
- COBOL-改進了對 OF 鏈接ADDRESS部分記錄的支持
- COBOL-增加了對 EXP EXP1 0 功能的支持
- COBOL-改進了對使用字帖REPLACE語句的支持
- COBOL-改善多維度欄位存取以支援已簽署值
- MF COBOL-增加了對可變格式順序文件的支持
- IMS-改進了對IMSYML文件配置的讀取，以便可以使用環境變量
- IMS-處理指定段編號的其他方法
- IMS-增加了從程序啟動的事務調用IMS程序時的穩健性
- IMS-改善搜尋條件SSA建置，以便在未提供隱含區段長度的情況下將WHERE子句的目前長度納入考量

- IMS-改進了對IMSYML文件配置的讀取以允許使用環境變量
- 改進了對中VALUE條款的支持 NumericEditedType
- 改進了對字符串連接的支持，以處理要連接的第一個字符串為空，空或空格時的情況

AS400

新功能

- 增加了對 Table 組件內的分頁支持; 項目可以使用此功能來減少響應時間和大小，當加載了大量行的 Table 組件
- 增加了對 AS4 00 應用程序SQL查詢的庫支持; 由於庫被轉換為現代應用程序中的分區，因此我們調整了運行時以相應的重寫查詢
- RPG-增加了對SQL查詢QTEMP庫的支持
- RPG-在CONVERT函數中新增編碼以處理空白輸入值
- RPG-增加了對% HOURS、% 和% MINUTES SECONDS 函數的支援
- CL-添加了該CHGPFM命令
- CL-增加了對CRTDUPOBJ命令中的 * FROMLIB 關鍵字的支持
- CL-為超過 9 個字符的表名稱添加了對表和分區創建的支持
- CL-為DLTF命令添加了對刪除子文件夾中的平面文件的支持

改善

- 屏幕-改 ErrorMessage 進與特定字段綁定並添加到 ArrayMessageLine
- 屏幕-改進錯誤信息光標
- 屏幕-改進 ArrayMessageLine 為不包含在標籤順序
- 畫面-改善 AS4 00 畫面錯誤訊息陣列的顯示
- SQL-改善關閉時提交交易的游標支援，以避免在建立磁碟分割時發生鎖死
- CL-增加了對該 PgmCall 命令的支持並改進了QCMDEXC不支持的模式
- CL-改進了對處理命CHKOBJ令的支持 OBJTYPE PGM
- CL-改進了處理庫CPYF和分區的其他 CL 命令的多庫支持
- CL-增加了在CALLPGM命令中傳遞一個程序名變量的支持
- CL-處理對象類型的默認類型的情況

- CL-增加了對命令的多庫支持 CRTDUPOBJ
- CL-增強了對多個命令的數據庫連接處理
- CL-改進了在找不到文件或目錄和 CPF 0000 監視器消息時處理案例的支持 RMVLNK
- CL-改CLRPFM進了刪除記錄時考慮庫
- CL-CPYF-改進了支持QTEMP庫的命令， FmtOpt (*NoChk) 參數和控制字符
- CL-修正了和CPY指令中引號和遺失參數RMVLNK的處理
- RPG-增強的變量範圍; 現在處於工作範圍而不 DataArea 是鏈接範圍
- RPG-改進了在沒有交易的情況下運行的DAO讀取查詢以避免死結
- 通過MSGQ在數據庫查找上添加修剪來增強 MQ 消息查找
- 刪除數據庫連接支持不必要的事務聲明
- 改善了石英工作狀態在發生異常情況下的更新
- 添加了在未初始化指標數組時處理情況的支持

橫向能力

新功能

- Redis 的-增加了全局 Redis 的配置為所有 Redis 的緩存
- 新增工作階段追蹤功能，可透過將資料保存至 Redis 來儲存工作階段追蹤資訊 (工作階段 ID、關聯的使用者名稱、建立時間戳記和節點 ID)
- 通過屬性添加了運行時解析 groovy 文件的臨時位置配置`tempFilesDirectory`; 還增加了指定是否通過YML屬性在應用程序啟動時清除臨時文件文件夾的內容的YML功能`cleanTempFilesDirectoryAtStartup`

改善

- 增強對公用程式資料來源之連線集區實作組態屬性的支援
- 根據條款和ADVANCING子句的使用改進了對打印機模式和ANSIWRITEBEFORE回車控制的支持
- 對現代化項目的前端應用程序更新的角度版本
- 增強的秘密管理器URL語法構建 DB2
- 增強了 DataUtils. compareAlphInt 添加對尾隨空格支持的方法
- 改進了SQL對 blob 類型輸出的支持
- 通過後/腳本端點添加了工作觸發器的強大性

現代化工具 4.2.0 版

ZoS

新功能

- CICS-增加了對解析WEBCICS命令的支持
- CICS-增加了對命MONITOR令轉換的支持
- CICS-增加了對解析CICS命令的支持 SEND MRO
- COBOL-增加了對解析 NO REWIND 語句的支持
- COBOL-增加了對CICS命令UCTRANST中選項數字類型的支持 SET TERMINAL
- COBOL-在 I-O-中添加對MULTIPLEFILE條款的支持 SECTION
- CSD-增加了對多個CSD文件的轉換支持
- CSD-增加了對從多個文件生成 jicsFileAix .json 的支CSD持
- IDCAMS：支援建立相對記錄資料集 (RRDS)

改善

- 改善了計算SQL遮罩時的效能
- COBOL-改進了無用RESERVE子句的解析 FILE-CONTROL
- COBOL-改善SECTION和的剖析功能 CLASS
- COBOL-改善DFHRESP處理能力
- COBOL-EXIT PARAGRAPH 通過執行的增強支持
- IMS-改善使用雙括號指定區段名稱的支援
- IMS-在調用SCHD和TERM調用時豐富了狀態代碼的生成
- COBOL-改進了 DEPENDING ON 字段的生成
- COBOL-改進了 TO_ TIMESTAMP DB2 內置功能的轉換

AS400

新功能

- 添加了對轉換字母數字字段的支持，如CHARSQL腳本

- COBOL400-增加了對程序DATABASE描述文件的支持

改善

- DDS-改進了對ALIAS名稱的支持
- 增強了對沒有初始值的 float 類型的支持
- COBOL400-改進了簽名分區類型的大小計算

橫向能力

改善

- 改善錯誤 ID 報告DDS和SQL剖析
- 改進了條件分支上的代碼生成
- 改善天氣報告產生的效能

發行版本公告 4.1.0

發行日期：二零二四年五月三十一日

此版本的 AWS Blu Age 運行時和現代化工具專注於性能和安全性。此版本中的一些主要功能和變更包括：

- 轉換和性能：為了允許具有大型代碼庫（+ 50M 行代碼）的項目成功轉換，我們優化了整個轉換機制的性能和內存佔用。
- BAC/JAC：安全 AWS 是最高的優先級。使用 AWS Blu Age 現代化的應用程式必須符合安全標準。我們對管理主控台 (BAC) 和 BluSam 管理主控台 () 進JICS行了一些重大升級，以使其更安全：JAC
  - 更新了應用程序角 V17。
  - 除了 AWS Cognito 的原生支援外，我們還新增了一般支援，可讓客戶使用自己選擇的身分識別提供者，提供更多彈性。OAuth
  - 使用適當的標頭配置和擴展安全功能。
- AS400-用於數據庫鎖定機制的多節點支持。提供插入共用和外部快取伺服器 (Redis) 的可能性，以便在多個執行個體上執行批次應用程式，例如受管理的 AWS 大型主機現代化。

此版本的 Blu Age 運行時已通過以下堆棧進行了測試。其他版本也可能兼容。

元件	已測試版本
Java	Java 17
表示層	節點 JS 18.18, 故宮 9.8, 角度 16.1
服務層	彈簧靴 3.2.5，彈簧核心 6.1.5，彈簧啟動器 4.0.0
持久層	甲骨文後 SQL 14
應用程式伺服器	阿帕奇雄貓 10.1.17

如需有關此發行版本中所含變更的詳細資訊，請參閱下列各節。

執行階段版本 4.1.0

ZoS

新功能

- 已新增動態提OAuth2供者處理的組態。引入 SECRET _ OAUTH2 _ _ PROVIDER NAME _ KEY 來指定提供者。更新秘密擷取方法來處理多個提供者。確保安全地從中 AWS Secrets Manager檢索密碼。
- 已新增對DB2SSL屬性的支援， AWS Secrets Manager 以便您可以定義SSL憑證 (sslTrustStore位置) 和密碼 (sslTrustStore密碼) 來解除鎖定金鑰儲存檔案。
- 新增對外部業務資料來源的支援。
- JCL-支援批次重新啟動的檢查點機制。
- JCL-增加了對DCB參數記錄大小和支持RDW。
- JCL-為生成的臨時文件添加了動態文件夾名稱配置。
- REDIS-在 Redis 配置中添加了池配置。JICS
- REDIS-增加了數據庫索引 Redis 的配置目錄和JICS。
- BatchScript -為執行程式執行新增步驟名稱的傳播。
- CICS-增加了對ADDRESSSET命令的支持。
- CICS-增加了對PURGEMESSAGE和的支持JUSTIFY。

改善

- JCL-INFUTILB-增強了根據YML屬性禁用 null 指示器的支持。
- JCL-INFUTILB-改進了CHAR/BPCHAR數據類型的支持。
- JCL-ICEGENER-支援將多行輸入串流複製到檔案中。
- JCL-IEBGENER-改善處理從變數區塊轉換為固定區塊檔案的支援。
- JCL-DFSORT-改進了操DATE作時對多位數參數的支持。
- JCL-DFSORT-增加了對 INCLUDE = ALL 子句的支持。
- JCL-改善SORT公用程式的支援，以處理輸出中的BDW欄位。
- JCL-改善對 DD 串連的支援。
- JCL-改進了對輸入流的支持。
- JCL-DSNUTILB-改善 NULLIF () 陳述式的支援。
- JCL-INFUTILB-增加了使用NOPAD選項卸載數據的支持。
- JCL-INFUTILB-增強了對中當前日期的支持INFUTILB。
- JCL-使用文件之前添加了文件存在和大小檢查。
- JCL-GDG-改善的子目錄處理。GDG
- MQ-改進了JMS實現中的連接打開。
- MQ-改進了 XA 數據庫GET消息的數據長度設置。
- MQ-分解的CMQV標準字帖，以防止編譯錯誤和重構用途。
- BluSam -改善對不存在資料集之刪除要求的支援。
- 改進了對ALLOCATE語句的支持。
- 改善 TS 命名的穩健性。QUEUE
- BatchScript -增強工作重新執行中先前步驟傳回程式碼的保留功能。
- 資料集-改善檔案存在且暫存時的檔案存在檢查。
- 資料集-改善尋找要刪除的GDG檔案時的並行性。
- 資料集-新增取得資GDG料集記錄大小的支援。
- CICS-改進了對INQUIRETASKLIST命令中SUSPENDED選項的支持。
- CICS-改善LOADSET使用 OF 陳述式ADDRESS的支援。
- CICS-改善未處理的CICS引數REMOTESYSTEM時CICSINQUIRE。
- CICS-增強對使用 OF 關鍵字定義指標處理SET選項的GETMAIN命令支援。
- JICS-透過新增交易狀態檢查，改善 jicsXAPrepare () 方法的穩健性。

- JICSXA-增加了事務狀態和增強事務線程終止的檢查。
- BAC-在客戶端增強基於角色的身份驗證，並重構/集中所有呼叫。API
- BAC-實施了阻止公共訪問BAC和JAC基於配置的功能
- BAC-升級依賴關係：角 17。
- BAC-改進了與 OAuth2- StateFarm/的安全集成FIDIS。
- BAC-產生DDL增強的休眠功能。
- BAC-改進了導出數據集機制。
- JAC-更新到角 17 和報告所有細節的工作從BAC（ROLE，管理員 CONFXSRF，註銷）。
- COBOL-增加了對CHAR和 ORD-MIN 功能的支持。
- 增強功 FileFactory 能可保持目錄記錄大小MOD處置。
- 使用JICS交易啟MDC用記錄功能。
- 已改進 SQLCA > 為SQLSTATE產生臨機操作結果集的預存程序產生。
- 改進了與上次 Spring 升級相關的任務計劃的支持。

AS400

新功能

- 使用 Redis 新增對資料庫記錄鎖定的多節點支援。
- 增加了BINARYCHARACTER對DDS類型的支持。
- CL-增加了對自訂報告檔案產生的支援。
- RPG-支援主要/次要RENAME檔案的關鍵字。

改善

- 改進了使用JOIN子句處理CTID列的數據庫支持。
- 改進了多個DSPATR（PC）的光標位置。
- 改進了讀取異常的日誌記錄。
- 改進了 Quartz 作業記錄功能，以將作業屬性納入MDC。
- 改進了對 AS4 00 幫助屏幕的支持。
- CL-改進了對RMVJOBSCDE命令的支持，以接受帶有尾隨空格的條目編號。
- CL-改善使用一般工作名稱移除作業排程之RMVJOBSCDE指令的支援。

- CL-改進了對按表鍵排序記錄的SAVOBJ命令的支持。
- CL-改進了為數據庫查詢建立新連接的CPYF命令的支持。
- CL-改善了在佇列訊息中插入查詢訊息的功能SNDPGMMSG。
- CL-改善工作佇列配置以指定預設工作佇列。
- CL-改進了支援資QTEMP料庫和RCDLEN參數的CRTPF指令。
- CL-改進了對CHKOBJ命令的支持-檢查與庫的分區。
- CL-改進了RTVMGS發送 CPF24 07 和文件/ID 未找到CPF2419時。
- CL-改進CPYTOIMPF和CPYFRMIMPF解釋繼承格式化參數。
- CL-增加了對OVRPRTF參數的支持USRDTA。
- CL-改進 CPYTOIMPF CL 指令以建立新連接對以避免關閉現有結果集。
- CL-改進，CHGDTAARA以便在更新內容時不再修改數據區域長度。
- CL-改進了 ClCommand 數據庫連接處理。
- 優化前端和後端之間的交互。
- COBOL-更新了轉型以處FILLER理字筆。
- 改進了發送到前端的自定義消息的其他消息信息顯示。
- 更新了應用程序中選擇器的默認值。
- 改進了 split-dynamic-field 顯示中的文本分割。
- 改進了多次寫入後跟讀取的錯誤消息的顯示。

橫向能力

新功能

增加了對OAuth2提供程序密鑰的動態配置的支持。

改善

- 打印-改進了用於處理引號的QCMDEXC參數支持和改進的報告名稱形成
- 改進了對上的分隔語法的支持 RecordAdaptable。
- 增強 InspectBuilder 錯誤記錄功能，可新增關於來源字串的內容
- DataSimplifier -增加了對情感的堅固性 ByteArray 。
- 使用新的執行階段屬性增強MDC記錄。

現代化工具 4.1.0 版

ZoS

新功能

- 增加了對多個CSD文件轉換的支持
- COBOL-增加了對CICSALLOCATE聲明的支持。
- COBOL-在ADDCORRESPONDING陳述式中增加了SIZEERROR對 ON 的支援。
- COBOL-增加了對 EXITPARAGRAPH.

改善

- COBOL-改進的支持-INC 字帖。
- COBOL-增強了對FILLER初始化的支持。
- COBOL-改善了比喻值比較的支援。
- COBOL-增強對WHENANY缺少中介程式碼區塊的連續WHEN子句的支援。
- COBOL-改進了對比喻常量的支持。
- COBOL-改進了對打包類型大小計算的支持。
- COBOL-已改善KEEP的SPOOLCLOSE未處理CICS引數。
- COBOL-改進了 TEST-NUMVAL 功能的生成。
- COBOL-改善INSPECT架構支援的 Java 產生引數。
- CICS-改進了對定義的支持DFHCOMMAREA。

AS400

新功能

- RPG-新增錯誤捕捉機制來產生 (不完整)，DDS因此不會封鎖程式產生。
- 增加了對INCLUDE文件描述規範關鍵字的支持。

改善

- RPG-改善完全免費剖析功能。

- RPG-增加了穩健性與錯誤捕捉。
- RPG-改進了具有導出關鍵字的字段/D 的初始化。
- RPG-改進了處理指示器的DAO操作。
- RPG-處理的預設值PERRCD與CTDATA。
- RPG-升級 Free 剖析RPG器，以記錄每個剖析規則的唯一錯誤。
- PRTF-處理PRTF和之間的名稱衝突JRXML。
- COBOL-改善對LIKE關鍵字的支援。

橫向能力

改善

- 為錯誤 ID 增加了穩健性 API
- 大型專案轉型的效能最佳化。例如：略過封鎖的檔案、重複使用 Blu Insights 中的分類，以及更好的記憶體配置逾時。
- 優化COBOL/PL1轉換期間的內存佔用。
- 修正CVE了第三方（jQuery 和引導）。
- TC 中的受管理 timeoutParser 選項
- 改進了SQL查詢上的多個空格重寫。
- 改進了具有靈敏度屬性的只讀光標。

發行版本公告 4.0.0

發行日期：二零二四年四月八日

有關如何從 AWS 藍色時代運行時 3.10.0 遷移到 4.0.0 的說明，請參閱。the section called “從 3 月 10 日移轉至 4.0.0 版”

此版本的 AWS Blu Age 運行時和現代化工具專注於升級關鍵依賴關係和支持的技術，同時提高多種功能的性能。此版本中的一些主要功能和變更包括：

- • 從春季啟動 2.7 升級到 3.2.4，春季核心 5.3 升級到 6.1.5，Tomcat 9.0 升級到 10.1.17，以通過使用正在積極修補和維護的版本提供改進的安全性，性能和可維護性。
  - 在前端應用程序上延遲加載，以構建具有 2000 多個屏幕的更快的大型項目，並將顯示初始化從 10 秒減少到 300 毫秒。

- Support 在前端應用程序上DBCS顯示。增強雙位元組字元的支援，以提供處理雙位元組和單位元組字元的新字型、防止雙位元組欄位中的單位元組輸入，以及處理混合雙位元組和單位元組字元的欄位。
- AS400 Online 應用程式的執行緒監控功能，可執行具有並行化功能的 AS4 00 個應用程式。
- 透過新增可設定的機制來預先 RunUnit初始化程式內容，減少載入舊有複雜性中固有的複雜結構所造成的影響，藉此改善前後關聯和初始化

此版本的 AWS Blu Age 運行時通過以下堆棧進行了測試。其他版本也可能兼容。

元件	已測試版本
Java	Java 17
表示層	節點
	故宮九
	角度的 16.1
服務層	春季靴子 3.2.4
	彈簧核心 6.1.5
	春季站 4.0.0
持久層	波斯特格雷SQL引擎 14
	甲骨文 21
應用程式伺服器	阿帕奇雄貓 10.1.17

如需有關此發行版本中所含變更的詳細資訊，請參閱下列各節。

執行階段版本 4.0.0

ZoS

新功能

- 增加了對包括語句 '-INC CPYNAME '的支持。
- CICS-增加了對PUSH/POPHANDLE聲明的支持。
- COBOL-增加了對「TODYNAMIC」ASSIGN 的支持。
- 增加了對DB2UNLOAD使用的支持INFUTILB。
- 在INREC語句中添加了SEQNUM對關鍵字OVERLAY的支持。

改善

- SORT-新增對排序字串文字 C'... '中的特殊字元 (括號和星號) 的支援。
- SORT-改進了對 OUTFIL NOMATCH-（..）參數的支持。
- SORT-支援資SYMNAMES料定義。
- SORT-改善 TO= 和 LENGTH = 引數的處理。
- SORT-改善處理方MOD式。
- SORT-增加了對 HIT = NEXT 參數的支持。
- 增強功ICEGENER能，可增加對特定輸出檔案編碼的支援。
- INFUTILB-增強了對 WITH UR 子句的支持。
- INFUTILB-當為假時 writeNullIndicator ，增強了對卸載的支持。
- DSNUTILB-當關鍵字位於可SQL選NULLIF關鍵字之後時，增強了加載步驟的穩健性。
- DSNUTILB-增強對隔離欄名稱的支援。
- DSNUTILB-支援將空白檔案載入資料表。
- DNSUTILB-新增對DNSUTILBSYSDISC檔案MOD配置的支援。
- IDCAMS-增強的評論支持。
- JCL-增加了對雙引號列的支持 LoadTask。
- JCL-增強了關於白色步伐移除的UNLOADSQL查詢處理。
- JCL-當處理中發生異常以確保JSON格式時，Groovy 腳本的增強響應。
- JCL-改善 DISP = NEW 和 DISP = OLD 情況下的檢查檔案配置。
- JCL-增強支援以處理GDG基本名稱中具有特殊字元的多GDG代參考。
- JCL-增強了加載虛擬文件的支持。
- JCL-增強對 tempFilesDirectory YML參數的支持。
- JCL-改善需要在字串元素中逸出雙引號時的JSON傳回。
- JCL-增強支 FileUtils 持GDG基本名稱。

- JCL-用於執行DB2多個查詢的增強DSNTEP程序。
- 增加了對春豆的支持。
- 增強SQLConverter以避免糾正錯誤的日期。
- 已改善的 JicsTimeBuilder 處理YYYYDDD。
- 允許自定義罐子可以從常規訪問。
- IMS-增強了IMS數據庫實現中跨記錄的導航。
- IMS-增強，能CBLTDLI夠啟動程序使用清除。
- IMS-DFSRRC 00 能夠將參數從常規傳遞到後端程序。
- 新增JICS了對未透過transactionRunner.
- JICS-通過使用可配置的緩存提高性能。
- BluSam -新增開啟 BluSam 時停用暖機的支援，以增強大型資料集的效能。
- BluSam-改進了常規 BluSam 數據集的刪除/重命名行為。
- BluSam -增強了記錄操作的性能。
- 用於確定字符串是否為低值的方法的增強型數據增強器。
- 增強了對數據包十進制和排序順序問題的支持。
- 使用 AWS Secret 增強了DB2作為主要資料來源的組態。
- 增強 FileSystem API以顯示檔案狀態。
- 增強 DynamicFileBuilder 讀取串流輸入lineSeparator。
- 當處理 0 個字符集時，用於確定字符串是否為低值的方法的增強數據增強器。CUSTOM93
- SQL-改善SQL預存程序輸出處理。
- SQL-改進了具有別名的多個表的 lambda 映射。
- COBOL-改進了來自語句LENGTH的支持。
- COBOL-增加了對TRANSFORM聲明的支持。
- COBOL-支援 9 個新的數學函數。
- COBOL-改進了對 INTEGER-F-DAY FUNCTION 的支持。
- COBOL-改進了對涉及具象值的 88 級支持。
- COBOL-改善SETADDRESS陳述式的轉換。

AS400

新功能

- 已移除重複的指標實體。
- 增加了對DBCS角色的支持。
- 介紹了子文件記錄控制HELP關鍵字的處理。
- 添加了配置參數，以切換管道字符上的列名大寫和拆分註釋列內容。
- 增加了對使用 0x0c 作為打包類型字段的最後一個半字節的支持。
- RPG-已處理使用 ExtProc ('system') 宣告的原型。
- CL-處理了 cl-command RMVMSG + 的 CLEAR '參數引入了內存中的非程序消息隊列。
- CL-處理通用語句被傳遞給 SBMJOBCMD （ ）調用。
- CL-增加了命令STRCMTCTL和ENDCMTCTL。修改鎖定機制和清理事務和鎖定。
- CL-增加了對CPYTOIMPF命令RCDDLM參數的支持。
- CL-增加了在SAVOBJ命令填充零的處理。
- CL-已新增包含在的OBJ參數限定名稱中的程式庫處理RTVOBJD。
- CL-增加了對CPYTOIMPF指令參數STRDLMSTRESCCHR、和RMVBLANK的支援。
- CL-增強功RTVMGS能，可在找不到檔案/ID CPF2419 時傳送 CPF24 07。
- CL-改進的RCVF命令來接收來自任何提供的庫DEV參數記錄。

改善

- 更改了 Blue4iv 任務執行程序的默認值，以允許默認情況下更好的擴展。
- 參數助手修改為轉換字符串列表和 ElementaryRangeReference 字符串。
- 增強功CTID能可處理中不存在的欄POSTGRE。
- 增加了堅固性以支援使用者空間 API「」QUSPTRUS。
- 增加了對用戶空間APIsQUSRUSAT和QUSCUSAT.
- 增強了對用戶空間API（QUSPTRUS）的支持，沒有錯誤代碼。
- 增加了對使用石英進行 CRON Job 排程的支持
- 增強對RPG程序週期的支持。
- 改善了交易管理。
- 已改善相同交易中承諾控制下檔案的記錄鎖定。
- 改進了子文件初始化的處理。
- 改進了消息行滾動指示器的顯示。

- 防止透過資料佇列傳送的數字尾隨零。
- 改進了其他消息信息屏幕。
- 改進了JPA寫操作以考慮當前庫。
- 改善執行沒有參數的程式 ProgramJobExecutor 時的行為。
- 增加了將前端鏈接的參數直接傳遞給後端腳本的功能。
- 改善工作中繼資料的交易處理。
- CL-增加了對SECLVL中RTVMSG的參數的支持。
- CL-增加了空的實現CLRLIB.
- CL-改進了從數據庫和複製的CPYFRMIMPF支持CSV。
- CL-改進了CPYFRMIMPF實現以忽略額外的列。
- CL-改進CPYTOIMPF和CPYFRMIMPF解釋繼承格式化參數。
- CL-增加了參數 removeDecimalPoint 來格式化數值。SAVOBJ
- CL-改進了RCVF命令，以正確處理EOF條件。
- CL-RTVSYSVAL-實施 SYSVAL = QDATETIME。
- CL-修改OVRDBF命令以獲取字段作為默認表名。
- CL-RTVJOBA 不可用的參數值:USRLIBL。
- CL-在SNDPGMMSGMSGF參數中處理前導斜杠。
- CL-改進了命令中源文件中通配符的支持。DSPFFD
- CL-改進了RCVMSG和SNDPGMMSG中參PGMQ數的處理。
- CL-使RTVMSG參數MSG可選，以與舊文檔保持一致。

橫向能力

新功能

- 改進了在OPEN游標USING子句處傳遞參數時的功能。
- 效能：已改善前後關聯的初始化與 RunUnit 效能調整。

改善

- 改進了從INFUTILB實用程序UNLOAD命令轉儲低值的機制。

- 增加了對數據源秘密管理器當前模式選項的支持。
- 增強的運行時不考慮在不需要時在打開游標處傳遞的參數。
- 改進了數字字段的數字格式驗證。
- 改善高度 parallel 執行環境中的SQL診斷功能。
- 引入了代碼頁字節序列（FE FD）的 Unicode。
- DataSimplifier 效能最佳化-增強型指派陳述式。
- DataSimplifier 性能優化-改善數字類型初始化的默認值，以防止無 BigDecimal 用的使用。

現代化工具 4.0.0 版

ZoS

新功能

- 增加了處理異常結束PROGRAM的支持。
- 改進了生成AIX數據集的支持。
- COBOL-增加了對//JUSTIFIED字段ALPHANUMERIC條款ALPHABETIC的GRAPHIC支持。

改善

- 改進了TRANSCLASS資源定義的PURGETHRESH屬性處理。
- 改進了對數據定義和MOVE語句的支持。
- CICS-增強了對DELAY命令選項的支持MILLISECS。
- 改進了具有別名的多個表的 SQL lambda 映射。
- 改進了對父字段查找的支持。
- 改進了對COMMIT和ROLLBACK操作的 SQLCA sqlstate 設置。
- COBOL-通過評論過時的段落來增強解析
- COBOL-增強對REPLACING條款的支持。
- COBOL-支援數學函數ASINACOSLOGTAN。
- COBOL-在中增加了對多個AFTER語句的支持PERFORMVARYING。
- COBOL-增強對RENAMES（級別 66）字段的支持。
- COBOL-增強LENGTH的 OF 方法以取得陣列欄位中特定索引處的長度。

- COBOL-新增對陳述式中多個AFTER子PERFORMVARYING句的支援。
- COBOL-增強對RENAMES條款的支持。
- COBOL-增強對PICTURE關鍵字的支持。
- COBOL-增強對 88 級欄位剖析的支援。
- COBOL-根據表格資料項目的條件改善 goto。

AS400

新功能

- 添加了傳遞參數以指導前端 java 調用的功能。
- CL-改進了 % 的SST生成，包括支持 * LDA 與 CL→ Java。
- RPG-增加了對文件的程序描述記錄的DISK支持。

改善

- 改善顯示檔案，使用 "REFFLD" 關鍵字解析參考欄位。
- 改進了對顯示文件關鍵字的支持 SETOF-CSRLOC。
- 關閉後從承諾控制項中移除檔案。
- 由同一程序執行時，確保表上並發讀寫操作的一致行為。
- 已處理指派給的 SizePrefixedAlphanumericType子字串。
- 使用可變長度的字符串參數處理將數據結構傳遞給過程。
- 改善 onBlur 事件時無效數值的保留，以及僅針對有效欄位建立事件偵聽程式。
- 改進了屏幕上的錯誤消息，並突出顯示具有無效輸入的字段。
- 改進了對指示器條件的屏幕字段的處理。
- 使用鼠標滾輪啟用滾動。
- 增加了對幫助屏幕功能鍵的支持。
- 改進了對 split-dynamic-field 組件中長文本的支持。
- 改進了重命名記錄時對多記錄 LF 文件的處理。
- CL-改進了處理 LF 文件（視圖）的RTVJOBD命令。
- CL-改進了在多記錄 LF 上使用時的OVRDBF命令。
- RPG-處理程序定義與重新命名參數相同名稱的變數的案例。

- RPG-改善初始化簽署ZEROSbinaryInteger時 * 的處理方式。
- RPG-改善非本機 (參考) 變數指標的處理。
- RPG-改善ELSEIF陳述式之後IFxx陳述式的處理。
- RPG-增加了對原型定義字段的支持。LIKE
- RPG-改進了對由創建的LIKE字段的關鍵字的支持LIKEREC。
- RPG-改進了具有比喻的運營商的生成。
- RPG-改善陣列運算式 xxx (\ *) 的剖析功能，並在% 查閱中支援此功能。
- RPG-改 LookUp 進了具有高和等於（或低和等於）指示器的操作代碼。
- RPG-改善自由格式剖析。
- RPG-改進了遵循 i-card 記錄格式的名為常量的解析。
- RPG-改進了對類型INTEGER和的支持UNSIGNED。
- COBOL-在聲明中添加了DSPF格式的支持INDIC條COPYDDS款。
- COBOL-改進了解鎖轉換DISPLAY和生成的語法和語ACCEPT句。
- COBOL-增加了來回DISK文件的支持。
- COBOL-改進了DDS顯示文件支持程序。
- COBOL-增加了對LIKE子句的支持。
- COBOL-增加了對程序描述文DISK件的支持。
- COBOL-增加了對帶後綴的文件名的支持。

橫向能力

新功能

- 處理 Web 項目的地圖組件的延遲加載。

改善

- 改進了 java 生成的SQL指標參數。
- 改進了處理SETDB2語句中涉及的變量的能力。
- 改進了輸出是單個實體數組時，獲取的光標末尾的錯誤提高。
- Linux 中的管理路徑。
- 資料移轉器可管理弱點並移除未使用的相依性。

發行版本公告

此版本的 AWS Blu Age Runtime 和現代化工具專注於整個產品的核心基準升級和改進，以提高所有轉換和執行步驟的性能和穩健性。此版本中的一些主要功能和變更包括：

- 從 Java 8 升級到 Java 17 的版本，提高了安全性和效能，並允許客戶部署和運行以更現代語言實施的應用程序，並使用最新的第三方框架版本。
- 額外支援管理使用者或作業之間的大型共用記憶體空間，並在應用程式或執行個體重新啟動後儲存資料
- 使用分頁機制，可以逐步檢索記錄的子集，更快地訪問 Blusam 中的大型數據集。

如需有關此發行版本中所含變更的詳細資訊，請參閱下列各節。

執行階段版本 3.10.0

此執行階段是以 Java 17、彈簧 2.7 和角 16 為基礎。

ZoS

新功能

- Blusam-通過其中索引存儲和使用頁面加載分頁機制增加了對大數據集的支持

改善

- 增強的 DataUtils .compare 處理從字符串到數字的較低優先級轉換
- 增加了支持通過屬性檢查 ByteRange 是否使用不正確的YML值創建dataSimplifier。byteRangeBounds檢查
- 增強了 remove SOSI () 以支持 GraphicAlphanumericType 具有空字符的初始化
- 增加了作業操作和安GDG全狀態讀取的穩健性
- 布魯桑-增加了通過名為的新方法清除布魯桑數據集的 Ehcache 的支持。 CoreBluesamManager removeCache()
- Blusam-改進了常規 Blusam 數據集的刪除/重命名行為
- Redis-增強了解鎖數據集和清除記錄鎖定的支持
- JICS-改善失敗要求的錯誤訊息
- JCL-支援以點字元為基礎的 ControlM 變數串聯

- JCL-增加了對GDG文件寫ADVANCING（ADV）的支持
- JCL-刪除所有GDG文件後增強了對當前代號的支持
- JCL-增強了在數據集創建時從目錄中recordSize 讀取rdw/讀取的支持
- JCL-增加了在打開與數據輸出記錄大小的文件時更新資源對象（從 AbstractSequentialFile）的支持
- JCL-改善IDCAMS效能
- JCL-加入 "" 作為 "CHARCHARACTER" 的別名PRINTSTATEMENT來增強支援
- SORT-增強了對從 Blusam 固定長度數據集複製操作到具有可變長度的數據集的支持
- SORT-增強的排序語法來處理一些特定的語句

AS400

新功能

- 增加了對用戶空間及其相關支持 APIs
- 增加了對消息隊列TOMSGQ參數SNDPGMMSG和實現的支持
- CL-增加了對命FILE令和SPLFNAME參數的OVRPRTF支持
- CL-增加了用於使用CPYF命令處理相應分區表庫的支持
- CL-增加了在構建查詢時處理CHGCURLIB命令和考慮當前庫的支持
- CL-增加了對處理 cl 命令作為調用堆棧跟踪的一部分的支持

改善

- 改進 MessageHandlingBuilder 了更好地處理調用堆棧跟踪條目
- 改進了 contextPreconstruct 功能的 parallel 執行
- 改善由建立記錄時的顯示屬性 SFLINZ
- 改進SAVOBJ以允許處理多個輸出文件
- 通過將它們添加到從 Java 程序調用 programCallStack 時，改進了 groovy 程序處理
- 幫助模式的改進頂部定位檢測
- 提供 toPgm Q 參數時改進了 toMsg Q 功能 SNDPGMMSG
- 改進了預定義的消息和消息加載器的功能獲取
- 改進了內容中分隔符號字符的CPYTOIMPF處理
- 改進了READ記錄上的釋放鎖定

橫向能力

新功能

- 增加了對前端系統消息的翻譯
- 在中添加了一個新方法 ExecutionContext 來返回程序調用堆棧
- 無論實際環境如何，都可以設置行分隔符（用於數據簡化器）
- 增加了配置SQL模型JSON路徑的可能性

改善

- 改進了比較方法 DataUtils。 compareAlphInt（ ）涉及填充時
- 創建標誌以允許在光標查詢中對異常進行自定義行為
- 改善圖形LOWVALUES資料庫轉換

第三方

- 升級以緩解 CVE -2024-21634, -20 CVE 23-34055, -2023-6378,--5905484, -2023-46120, CVE -2023-6481, IN1 JAVA ORGSPRINGFRAMEWORKSECURITY CVE CVE CVE CVE

現代化工具版本 3.10.0

ZoS

改善

- COBOL-增加了對ABS功能的支持
- JCL-增強的變量範圍：附加到STEP而不是 JOB
- 增強了低/高值的游標參數注入
- 改進了CSD解析，尤其是遠程 TRANSACTIONS

AS400

改善

- 刪除了控制級指示器的空白檢查

- 增加了對外部名稱IMPORT/關EXPORT鍵字的支持
- 增加了對字段 % LEN 的支持
- CL-增加了對CLLE語言的新運營商的支持
- CL-增加了對嵌套 IF 支持
- COBOL-改進了與多個鍵一起使用時START命令的處理
- DSPF-改進了帶有記錄編號的光標位置處理
- DSPF-改進了已簽名數字，僅數字字段和大規模字段的格式
- DSPF-改善螢幕一般說明標題的決定
- DSPF-改進了對輸入/輸出規格的支持
- DSPF-改善驗證數值欄位時分組分隔符號的處理
- 改善對應輸出/記錄 DDS
- 改善印表機檔案REFFLT關鍵字解析參考欄位的能力
- RPG-增強對「ALL免費」語句的支持
- RPG-改善條件剖析功能，並新增CABXX無結果處理的支援 TAG
- RPG-改進了數字字段的輸入規範處理
- RPG-改進IF/條件內過程調用的處ELSEIF理 WHEN
- RPG-改善在 dspf 檔案上呼叫時READ指令的處理
- RPG-改善對參照不存在的檔案的支援 DDS
- 改善傳遞實體記錄格式名稱REFFLD時的處理
- 增加了使用「返回」作為數據庫列名的支持

橫向能力

新功能

- Oracle-有可能定義用戶而不是SYS存儲內置函數

改善

- Java 版本從 8 版升級至 17 版
- 已改SQL善叢集欄名稱的條件
- 從視圖添加了對 ORDER BY 子句的支持

發行版本公告

此版本的 AWS Blu Age Runtime 和現代化工具專注於跨產品的多項橫向增強功能，致力於提高高可用性架構的性能，以及將作業執行提升到一個新的水平的新功能。此版本中的一些主要功能和變更包括：

- 版本從 Angular 13 升級到 Angular 16，增加了安全性並提供新功能的訪問，以提高客戶的在線應用程序的性能。
- 在 AS4 00 中增加了對跨作業功能的支持，其主要亮度使作業可以在其中同步發送查詢消息，從而在現代化工作中進行解耦。
- Redis 的使用效能改善，包括連線集區最佳化、連線安全性高，以及升級的資料集鎖定機制。

如需有關此發行版本中所含變更的詳細資訊，請參閱下列各節。

執行階段版本 3.9.0

ZoS

新功能

- 排序程序：具有固定長度的更新VSAM輸入
- JHDBDB：添加了可配置的超時

改善

- 如果在文件連接中使用，則增強了對行分隔符進行流的支持
- 增強了打開連接的順序文件的支持。打開文件 DataSetIndex 後初始化
- 當 a NumericEditedType 受到數值影響時，增強對虛擬小數分隔符號的支援
- 增強對負值 NumericEditedType 的支持
- IDCAMS：現在使用 application-utility-pgm .yml 中定義的「編碼」屬性讀取SYSIN卡
- IDCAMS：更新了語法以支持語DEFINECLUSTER句中的FILE (..) 參數
- INFUTILB：支援DFSIGDCB引數以覆寫 DD 的DCB參數 SYSREC
- INFUTIL：增強對 "DFSIGDCBYES" 參數的支援
- 改進處SPLICE理巨大的輸入文件
- DFSORT：改善備註欄位處理
- DFSORT：支援 (有符號/無符號) 自由格式的數字格式 (SFF/UFF)

- SORT：新增對OPTIONPRINT和OPTIONROUTE陳述式的剖析支援
- SORT/ICEMAN：增加了對封閉式除法操作的支持（帶DIV運算符的字段）
- 增強了對CICSREAD使用通用密鑰的支持
- 功能 StringUtils .chargraphic 固定從圖形類型SOSI中刪除
- 增強 DataUtils. isDoubleByte編碼
- JCL：增強對臨時數據集的KEEP配置模式支持。系統會將處理方式變更為 PASS
- JCL：動態處理DCB參數
- JCL：增強了不正確值的SUMFIELDS輸出
- JCL: CommonDDUtils:: getContent 現在搜尋目錄 recordSize 中的
- JCL：在建立資料集時從目錄讀取 rdw/ recordSize 屬性
- JCL：增加了對 DCB = 的支持。 MYDD將 DD 的DCB參數複製到同一作業步驟中的另一個參數
- JCL：改進的記錄大小繼承系統
- JCL：增加了（Redis）獨家數據集鎖定
- Redis 的：增加了對獨立模式的SSL支持
- Redis：添加了帶鎖定的同步 Redis 鎖定計數
- Redis 的：Redis 鎖定支援的池參數
- Redis：使用 Redis 優化的元數據刷新
- Redis 的：改進了 Redis 的集群支持
- 改進 IO 模式打開鎖
- 改善資料集鎖定效能並清除未使用的鎖
- 增強解除註冊檔案期間資料集的路徑
- 改進了預取窗口緩存無效
- 增加了對線程安全實用程序數據源提供者使用的支持
- 增強型 datasetState 無效檢查
- 增強對不重新開啟已開啟的資料集的支援
- 增加了工作最終操作的穩健性
- 增強了對允許重複鍵的索引順序的支持
- 增強了對跳過列表序列化順序的支持
- 增加了對調試轉儲功能的支持，以幫助診斷索引順序問題
- 增強對中繼資料重新整理

- 增強了對 Blusam 批量讀取的支持

AS400

新功能

- 建立應用程式內容登錄
- Support 關DSPF鍵字CLRL（NO）Support 記錄鎖定監控
- Support 鍵控 DataQueue
- 批次作業的INQUIRY訊息 Support
- 為 00 添加了對程序描述的打印機文件的支持 AS4 COBOL
- 處理碼 RMVJOBSCDE cl 指令
- 改善RUNSQL/DLYJOB
- CHKOBJ：引發參數的遺留錯誤代碼 LIB
- SNDPGMMSG：支援字串參數
- RTVDTAARA：改善中的子字串 LDA
- DSPFD: FILE 參數支持為特定文件名添加
- RUNQRY：對於中的 sql 文件的 Support QRY PARAM
- CRTDUPOB：Support 在數據區域之間複製數據
- SBMJOB: 轉換指令以使用 JobQueueManager
- OPNQRYF：支援 Qtemp 程式庫
- CRTDUPOBJ：改善複製分割區內容的邏輯
- CRTDUPOBJ：已新增對檢視的 Qtemp 支援
- RTVSYSVAL：Support SYSVAL 值，QDATFMT在 CL 指令中
- CHKOBJ：增加了支持 OUTQ
- RTVJOBA：支持SWS參數
- SNDPGMMSG和RCVMSG：支援的其他參數 MSGFMSGFLIBMSGDTA、MSGTYPE、、KEYVAR、MSGKEY、MSGID

改善

- 改良的 WORKSTATION I/O 卡支援

- 改進了覆蓋先前消息的設置消息的處理
- 支持有關數組消息傳遞線的其他消息信息
- 改進了內部的獨立數組包裝器訪問EVAL，SorTA，模型工具
- 在線申請結束時改善DAOs清潔
- 增加了對其他日期格式的支持，並改善字符串輸入的處理
- SYSVAL通過添加系統值幫助程序類從 CL 命令解碼和構建參數來改進CVTDAT處理 SbmJob
- 從組件掃描中刪除軟件包。 gapwalk-cl-command
- 改進了對消息隊列預定義消息的支持 API
- 改進了在另一個程序中編寫的記錄的 retrieveSubfileRecord 支持
- 改進了對消息隊列的即時消息的支持 API
- 提交作業時改善本機資料區域的處理
- 伺服器啟 JobQueues 動時自動啟動
- 使用 applicationContext 配置解碼參數 SBMJOB
- 改進系統提供的錯誤消息
- 允RTVMSG許搜索嵌套子目錄中的 .properties 文件
- 處理綁定到壞/無效指針的實體的重置
- 改進 MessageHandlingBuilder 了顯示 msgId 和 MsgFile 命名為字符串 RCVMSG
- 改進了消息隊列的 withMsgFile名稱方法 API
- 改善資料區域鎖定機制
- RTVMBRD：Support 參數的小寫和大寫 FILE
- CRTDUPOBJ：改進了視圖的處理
- CPYTOSTMF：改善連線處理
- CPYF：從平面文件複製時處理目錄名稱的改進
- RCVF：正確處理常規和 Java 的 DEVRCDFMT/RCDFMT參數和轉換
- RCVF：處理後續呼叫並避免重設游標
- CPYF：支援從平面檔案寫入
- CRTDUPOBJ：添加了使用 Qtemp 庫對新 obj 的處理
- CHGDTAARA：將資料區域的最大長度從 256 個增加到 2000
- SAVOBJ：確定儲存的記錄是以插入順序
- RTVDTAARA：擷取的值 (不被修剪)

- CHKOBJ：當成員不存在時，返回正確的監視器消息
- RTVDTAARA：增加了對LDA子字符串的支持
- RTVDTAARA：返回到參數中指定的變量長度的空格 RTNVAR
- RTVDTAARA：支持開始和長度的整數參數，並支持最新的轉換格式
- CHGDTAARA：支援包含下限和上限的參數
- CHKOBJ：處理參VIEW數物件類型的值
- CHKOBJ：如果視圖存在，結果設置為 true，而不管成員

橫向能力

新功能

- 處理產生報告至 .txt 檔案
- 將 currentSchema XA 數據源屬性添加到秘密管理器
- 添加數據庫 .cursor.raise.alread.error 屬性，以使框架在已經打開光標打開時引發錯誤 502 YAML SQLCODE

改善

- 在 Amazon 包裝上為 AWS 藍色時代添加了間隙走動球 EC2
- 預設使用新的訊號處理常式範例
- 當處理方式為或時，新增對鎖定的支援 MOD OLD
- 添加了緩存來存儲數據庫日期時間模式
- 改進的檢查功能 PackedType
- 改善 DataUtils。 setTo 用於記錄的函數 VariableSizeArray
- 作為運行單元處理 MQ SYNCPOINT 選項
- 啟用框架SQLCODE在回滾事務上設置
- 根據引擎密鑰密鑰添加了自動驅動程序類名
- 程序/交易超時
- 訪問光標時回滾後恢復光標位置

第三方

- 升級蛇YAML，雷迪遜和 AmazonSDK，刪除 YamlBeans （緩解 CVE -2022-25857，-2023-42809，CVE2023-44487）CVECVE

現代化工具 3.9.0 版

ZoS

改善

- 增強了對 XML-TEXT 作為字符串類型的目標源的支持
- 增強STM到UML工作流程以支援 X/ (Y/Z) 分割模式
- JHDBDB：在任何數據庫更新之前接受ROLLBACK調用
- JHDBDB：ROLLBACK即使交易終止也接受（NOP）
- JCL：改進的步驟驗證功能
- SORT: 使用區域十進位負值處理SUM函數
- COBOL：增加對字串常值中單引號/雙引號逸出的支援

AS400

改善

- 通過添加前導零改進了內置函數 %editc 處理編輯代碼 X
- 改進了僅輸入字段初始值的處理
- 增加了操作鍵來幫助對話框
- 動態表的頁腳記錄出現在底部
- 沒有指定實際文件KEYPHASE的處理START命令 RECORD-KEY
- 增加了對浮動的默認值和: NumberUtils: 戰俘類型
- 增加了使用LIKE（IN）定義變量的支持
- 更新了FOR循環處理，以支持省略可選元素
- 更新了RPG解析以將記錄與CTDATA數組名稱相關聯
- 改進了CABxx語句指標的處理
- 支持COMMIT關鍵字可選參數
- 改善 LF 中的FORMAT關鍵字支援

- 具有高和等於（或低和等於）指標的託管LOOKUP操作代碼
- 處理在雙引號內聲明的 PF 鍵名
- 改進了 EDTCDE X 的處理，以不抑制前導零
- 改進了對打印機文件MSGCON中未生成未命名標籤的支持
- 字段CONTENT由多個數據結構共享
- 處理的ERRSFL參數與SFLMSG/SFLMSGID
- 完全免費 rpg 的 proc 聲明範圍之前改進了主代碼
- 添加了解析條件控制規範
- 改進了對數據保存器映射器中的 setErrSfl（ ）方法的支持
- 改進了內部創建變量的類型分辨率
- 改進了對 Z-ADD 操作碼的支持
- 改進了具有DFT值的常量字段的處理
- 提高程序狀態 ds 內對整數字段的支持
- 在ENTRY參數中處理指示器分配
- 改進了通過 ref/reffield 關鍵字傳播的關鍵字的過濾器
- 支援的未命名 DataArea 資料結構
- 改進了指針數據類型的處理
- 數組的處理元素用於在輸出字段中定義具有LIKE關鍵字支持數組訪問的變量
- 改進了對簽名數字的支持，僅顯示數字
- O 卡上支援的邏輯關係
- 測試用例 % CHAR 的字母數字
- 支援的控制規格關鍵字 main
- EDTCDE在打印機文件中有兩個參數
- 改善 FullFreeRPG剖析
- 增強了動態表格，以確保頁腳正確定位
- 增加了對具有ALL比喻常量初始化數字類型的支持
- 改進了引用相同物理文件的多個RPG邏輯文件的處理
- 改善現代屏幕中修改字段的檢測
- 與動態字段模態同步
- 改進了僅輸出簽名數字字段的處理

- 改善 WORKSTATION I/O 卡支援

橫向能力

新功能

- 數據遷移工具：添加了 ebcdicFilesWith VarcharIn VB 屬性，以允許在讀取VARCHAR字節時考慮 2 字節長度
- 實現了一個常見API的日誌錯誤
- 通API用的實現 BluAgeErrorDictionaryUtils 和使用記錄錯誤和/或信息 COBOL2ModelRPGCycleBuilder，定義 2Model 和 FieldsProcessor
- 改進了SQL語法以支持不同的隔離子句定義

改善

- 升級角版本到 V16
- 角度：從 6 到 8.9 升級的 AJV 版本

第三方

- 升級時髦的版本 2.4.15

發行版本公告

此版本的 AWS Blu Age Runtime 和現代化工具專注於產品中的多個橫向增強功能，以提高其質量和安全性，以及緩存性能的改進和命令支持在單個分佈中統一。此版本中的一些主要功能和變更包括：

- 版本從 Spring 2.5 升級到 Spring 2.7，提高了平台的維護支持，性能和安全性。
- 統一超過 82 個 CL 命令支援作為發行版的一部 over-the-counter分，以便於使用和部署先前使用 CL 指令碼的現代化應用程式。
- APIs提供更好的 Blu 數據SAM集操作和交互的新功能，例如對受管理服務的集成導入以及列出數據集元數據信息的功能。
- Redis 的效能改善和使用延伸，包括叢集模式下的可用性、高可用性資料擷取、標準化機密使用。

如需有關此發行版本中所含變更的詳細資訊，請參閱下列各節。

執行階段版本 3.8.0

ZoS

新功能

- 將鍵定義作為字符串處理 DynamicFileBuilder
- DFSORT：在 OUTFIL TRAILER1 + DFSORT 語法初始化中增加了對多項的支持
- C ommonDDUtils 工具：處理串流內資料中的記錄大小
- 索引文件：處理選GENKEY項

改善

- 外部化的 Blu SAM 加載服務在一個單獨的 jar 中
- 增加了設置存儲臨時文件的位置的支持
- 改進了多節點案例的共享緩存機制
- 共享緩存使用情況：IDCAMS驗證優化
- 改善嵌入式選擇的ROWID注射
- JCL：每個串流內作業程序現在都會在不同的 Groovy 檔案中產生
- 確保IDCAMSJCL信用卡覆蓋 card-demo-v 2
- 藍光SAM：使用多個實例 warmUp 時避免重複
- 減少快取水分的記憶體佔用
- 傑迪斯池配置支持
- 如果在文件連接中使用，則添加了要流的行分隔符
- Support EBCDIC 卡 + 塊評論（/*.../）在IDCAMS實用程序
- 數據庫支持查詢：在 level49 轉換中支持雙字節字符串 SQL
- DFSORT語法：實現 17 個控制語句 + 其中 2 個的集成（OMIT/INCLUDE）
- 增強GRAPHIC列獲取 INFUTILB
- Support 使用可變大小表格讀取檔案
- Support ZonedType 與 nibble 簽名，其中最後一個字節的第一位是 'E'
- DFSORT/ICETOOL添加了對 NOMATCH =(..) 參數的支持，如果一條記錄不匹配任何CHANGE查找常量
- Redis 的集群兼容性

- 根據 groovy 退出代碼處理 Job 狀態（失敗）
- 改善CICSSYNCPOINTROLLBACK支援。
- 預先擷取視窗以最佳化 Redis 快取使用情況
- JCL/GROOVY：繼承是從前一步的數據集RDW屬性時 DISP = (,) PASS
- 使用可變大小的數組處理數據的部分副本

AS400

新功能

- Support 顯示檔案的 I/O 卡
- Support 關DSPF鍵字ERRMSGID和其他消息信息 CHKMSGID
- Support 前端屏幕上的多個錯誤消息
- 在應用程序中添加或改進了 82 CL 命令的 gapwalk-cl-command支持

改善

- 改善對承諾控制DELETE和承諾READ控制的支援
- ConvertDate 內置 % 十二月內部
- 強制XSS安全標頭
- 改進了STM生成的穩健性和一致性（更好地處理：自由格式 rpg 中的延續行，小數部分的逗號，定義/聲明中的自由形式塊）
- 改善的 DataHolderMapper 一代
- 增加了堅固性並更改範圍 DataAreaFactory
- 改進了標籤鍵上的焦點移動
- 改進了賈斯珀報告生成的性能
- 改進的十進制顯示與填充 0
- 改進了對ROW/COL字段的支持 INFDS
- 改善畫面對修改欄位的支援
- 為生成的報告名稱和路徑添加了獲取器
- 改善資料佇列長度
- 改進了 Job 隊列的自動配置以匹配 Spring Boot 2.7 中的新標準
- 已改善多個並行工作階段的工作站

橫向能力

新功能

- Support 封裝無無效的資料公差
- 新增分頁/篩選列出資料集端點

改善

- 增強的ORACLE查詢轉換策略，以列與空字符串進行比較
- 處理BLOBDB2DSNTEP和INFUTILB實用程序。BLOBDB2現在已經現代化為BYTEA鍵入Postgres。
- 改進刪除光標的最後一個項目
- 增強對刪除RRDS文件的支持
- 改善 AWS 布魯桑秘密效能
- 改進了SQL框架中數據庫連接的處理
- 標準化的 AWS 多資料來源秘密管理員金鑰
- 效能迴歸修正
- 改進了檢查功能 PackedType
- 改進的處理 LOW-VALUE PackedType
- 用於認知連接的升級彈簧安全包裝
- 不在目標數據庫上應用代碼提交點編碼和解碼 DB2

第三方

- 春季啟動從 2.5 升級到 2.7

現代化工具 3.8.0 版

ZoS

新功能

- JCL：使用回車符「\ r」處理流

改善

- 改進了日誌記錄以防止在DIVIDESIZEERROR使用 ON 子句現代化時除以零
- JCL：增強對程序中呼叫程序的支援
- 當存在不明確的字段時，在FORMATTIMECICS命令中 Support OF 關鍵字
- JCL: 支援變數中的 â¥ 字元
- JCL：根據先前的步驟計算 RC
- 使用時比較字節而不PL1SUBSTR是字符串
- 改進從單一源代碼初始化多維數組
- 改進了COBOL在 IF 塊中涉及單個SQL查詢的解析

AS400

新功能

- 在 CL 中 Support 嵌套 IF 語句
- 改善對RPG自由格ENDDO式中陳述式的支援

改善

- 改進了對調節控制級別的支持
- 改善原型回報 LIKE
- 改進了對處理函數 % 月，% 年，% 天的支持
- Support 整個屏幕的幫助功能
- 處理比喻作為參BLANKS數傳遞
- EVAL使用「」運算符改進表達
- 沒有處理START命令 KEY PHASE
- 關鍵字處理的改進 LIKEREC
- 改善未命名分欄
- 改進返回無符號類型的過程
- 改進了對RESET操作（免費RPG），% CHAR 和 % DEC 內置插件的支持
- 改善內建功能% LOOKUPXX
- 改進了對沒有原型的過程LIKEDS關鍵字的支持

- 處理 Dim 關鍵字數組類型（VAR，AUTO）
- 改善的支援 XFOOT
- COBOL：改進了對RENAMES字段的支持
- CL：支持而（真）條件
- 使用LIKE關鍵字改進了獨立數組的處理
- 改善內建函數% INT
- 改進了RPG完全免費解析
- 改善連結中陣列的支援
- CL2GROOVY：Support 選擇聲明
- DSPF關鍵字「ERRMSGID」的改進
- 改進了使用前導零初始化字節的處理
- 數字欄位 authorizedValues 的改進
- 處理擴展器 H 的自由形式EVAL聲明
- CL 到 Groovy：Support 子字符串 LDA
- 改進了對記錄RESET的支持
- 改進了參考EDTCDE和EDTWRD的處理
- 改進了帶有字段的輸入字段映射 DDS
- 改進了對MOVEA字符進入 IN 數組的支持
- 使用LIKEDS關鍵字改進原型
- 改進了對DSPF關鍵字的支持 DSPATR
- 使用 +/-改進了 D 卡的解析
- 在程序調用中增加了堅固性
- 在現場解析過程中增加了穩健性

橫向能力

改善

- FrontEnd：模擬IME輸入的粘貼事件

第三方

- 春季啟動從 2.5 升級到 2.7

發行版本公告

此版本的 AWS Blu Age 運行時和現代化工具主要包括增強功能以更好地支持命令和實用程序，與 AWS Secrets Manager 集成的功能以及新的監視功能。此版本中的一些關鍵變更包括：

- 多個執行階段元件現在可以使用 AWS Secrets Manager 來增加現代化應用程式的安全性設定，主要與公用程式資料來源、Redis for TS 佇列、 BluSam 快取和鎖定有關。
- 監控端點，允許檢索交易，批處理和JVM指標，以優化資源使用和操作管理，例如狀態，持續時間，數量等。
- 支援 IBM MQ 呼叫RPG，以及增加和IDCAMS轉型涵蓋範圍JCLSORT的新功能。

如需有關此發行版本中所含變更的詳細資訊，請參閱下列各節。

執行階段版本 3.7.0

主題

- ZoS
- AS400
- 橫向能力

ZoS

新功能

- 通過使用 SQL like 語法改進涉及程序實用程序應用程序的解析查詢。(七、九四零一)
- 偏移時處理索引的可變大小數組 (V7-9904)
- Support INSERT SQL TIME 列DB2以 24：00 小時格式進入（V7-10023）
- Support 使用FORROWS和ATOMIC選項的陣列進行INSERTSQL查詢 (V7-10105)
- JCLSORT-增強 TranscodeTool OUTREC與支持IFTHEN（V7-10124）
- JCLSORT-在OUTREC命令中添加對DATE關鍵字的支持（V7-10125）
- JCL-增加對串流程序的支援 (V7-10223)

改善

- 標有 "PASS" 處置方式的資料集應該可在所有作業步驟中使用 (V7-9504)
- S JCL up SCHENV port 屬性 V7-9570
- SENDSupport 搭配CTLCHAR選項 (V7-9714)
- COBOL-處理ACCEPT陳述式中不同的行分隔字元字元集 (V7-9875)
- 避免多次復原 (V7-9958)
- 允許使用MOD配置在GDG檔案結尾附加 (V7-10031)
- 優化： putAll 重構（V7-10063）
- PutAll 重構：添加分頁（V7-10063）
- 使傑迪斯客戶端讀取超時可配置（V7-10063）
- UseSsl 支援獨立模式 (V7-10114)
- 成功開啟檔案EIBDS後的 Support (V7-10147)
- 檔案控制要求EIBDS後的 Support (V7-10147)
- 改善CICSSYNCPOINT支援能力 (V7-10187)
- BluesamRedisSerializer：問題與 metadataPersistence (V7-10202)
- 對於 TS 隊列 Support 雷迪斯 AWS Secrets Manager（V7-10204）
- Sup JCLBCICS port 自訂 DD 名稱大小 (V7-10224)
- 在IDCAMSDELETE語句中添加對絕對路徑的支持 (V7-10308)

AS400

新功能

- 實作 AS4 00 個螢幕的說明功能 (V7-9673)

改善

- 中的記錄數目 INFDS (V7-9377)

橫向能力

新功能

- Support 運行時EC2將日誌發送到 Amazon CloudWatch （D87990246）
- 已新增用來擷取有關批次、交易和指標的新端點 JVM (D88393832)

改善

- Support 公用程式 PGM 的資料來源 AWS Secrets Manager (V7-9570)
- 增加了對下列項目的 Db2 支援 DSNUTILB DISCARD (V7-9798)
- Support 在默認SYSPRINT和SYSPUNCH文件中寫入記錄器而不是默認系統輸出流（V7-10098）
- Support BluSam Redis 的緩存和 AWS Secrets Manager 鎖定連接屬性（V7-10238）
- Support 於 Db2 XA AWS 秘密的SSL連線支援 (V7-10258)
- 已更新IDCAMSREPRO和的中繼資料 VERIFY (V7-10281)
- 改善IDCAMS異常結束傳回程式碼管理 (V7-10307)

現代化工具 3.7.0 版

主題

- ZoS
- AS400
- 橫向能力

ZoS

新功能

- PLI-改善陣列橫截面和二維陣列的指派 (V7-9830)

AS400

新功能

- 控制水平指示器的處理 (V7-9227)
- Support EXTNAME 參數 * INPUT (V7-9897)
- 增強的轉到重寫:Support 位於SELECTOTHER陳述式中的標籤 (V7-9973)
- Sup REFSHIT DSPF port 關鍵字 (V7-10049)

改善

- 處理文件描述關鍵字的改進EXTIND（*INUx）（V7-7404）

- 改善SQLDDS檔案轉換功能 (V7-7687)
- 不再為 AS4 00 個檔案產生檔案物件 (V7-9062)
- 改善檔案描述關鍵字的處理方式 EXTDESC (V7-9268)
- 改善了CHAR內建百分比的處理能力 (V7-9311)
- 改進了對上一筆記錄的下一頁支持，而不SFLEND（V7-9322）
- 改進了對前綴數據結構的支持（V7-9436）
- Support 以% 定義的尺寸 SIZE (V7-9472)
- Support 處理雙引號內宣告的 PF 欄位名稱 (V7-9557)
- 改進的文件操作-不區分大小寫 (V7-9785)
- Support 初始化為 * 的欄位 USER (V7-9806)
- Support 在 AS4 00 的COMP類型 (V7-9840)
- 改進了 COBOL4 00 解析（不是）InvalidKey （V7-9922）
- 改良的SCAN操作處理能力 (V7-9971)
- 改進了GOTO操作碼的支持（V7-9973）
- 改良的EXCEPT操作處理能力 (V7-9977)
- 改進的前綴支持（V7-10000）
- Support 中的 MQ 通話 RPG (V7-10007)
- 改善% LOOKUP 內建 (鍵控陣列資料結構) (V7-10022)
- Support 關閉 * 全部操作 (V7-10036)
- Support UPDATE AS ROW CHANGE SQLDDS 陳述式 (V7-10051)
- 處理常值類型長的改進 (V7-10073)
- 改進的RPG語法（使用關鍵字INZ作為子程序的名稱）(V7-10074)
- 改進了RPG語法以支持具有空小數部分的數值（V7-10077）
- 改進了對 CL 和外部文件之間共享字段的支持（V7-10081）
- 改善對DDS條件式指標的支援 (V7-10084)
- Support 含COBOL程式的DDS二進位類型 (V7-10100)
- 改善與連結的名稱衝突 (V7-10109)
- Support 混合主要和出口程序（V7-10112）
- 改善子程序 DataStructure 中的支援 (V7-10113)
- 已改善的支援 CLEAR (V7-10126)

- 改善對 DO 迴圈的支援 (V7-10134)
- SQLTYPE在完全免費的 Support RPG (V7-10151)
- 改善DDS關鍵字的條件剖析 (V7-10155)
- 改良的DSL一代 (V7-10163)
- 條件為二進位運算式 processIndicators 時的改善。(V7-10164)
- 改進GOTOs與其他條件（V7-10168）
- Support 類型時間和時間戳DSPF（V7-10173）
- 改進了繼續行的解析DDS（V7-10183）
- COBOL對於RENAMESFLD的支援 RECORD (V7-10195)
- 改善DSPF欄位上的條件式指示器剖析 (V7-10221)
- Support DDS 關鍵字剖析 NOALTSEQ (V7-10288)
- Support 說明功能表和隱藏欄位 (V7-10314)
- 改善DSPF說明關鍵字完整性檢查 (V7-10328)
- 不再傳播參考欄位上的所有關鍵字 (V7-10347)

橫向能力

新功能

- 資料移轉程式-處理CLOB資料 (V7-9665)

改善

- 透過將JCL內容SCHENV從定義傳播JOB至PROCGROOVY定義 JobContext (V7-10225)
- FrontEnd -在沒有邊框的情況下調整窗口大小（V7-10358）

發行版本公告 3.6.0

此版本的 AWS Blu Age Runtime 和現代化工具為 ZoS 和 AS4 00 舊版遷移提供了新功JCL能，主要面向擴展CICS支持機制，補充功能，優化並發和大批量功能的性能以及添加功能。 multi-data-source 此版本中的一些關鍵變更包括：

- 增強了JCL動態文件處理，擴展當前語句和串聯的數據集的管理，在單個塊中執行多個語句以及從批處理到程序的數據傳輸。

- 增強對多個CICS指令的支援，包括查詢多種CICS資源類型。
- 使用 Blu Age 運行時實用程序時具有不同數據庫的功能，最適合業務數據分佈在多個來源的情況下。

如需有關此發行版本中所含變更的詳細資訊，請參閱下列各節。

執行階段版本 3.6.0

主題

- ZoS
- AS400
- 橫向能力

ZoS

新功能

- JCL- DynamicFileBuilder -增強型檔案處理程式管理功能 (V7-9408)
- 呼叫INFUTILBUNLOAD公用程式時，某些內建SQLDB2函數的增強格式轉換 (V7-9554)
- 增強型PLI多維度陣列指派 (V7-9592)
- 系統輸出重定向到文件的處理 (V7-9992)

改善

- 為下列項目新增預存程序的觸發程序 DB2 RDBMS (V7-9155)
- SORT處理轉換為PDF格式 (V7-9286)
- JCL/GROOVY-增強REPRO聲明以支持DUMMY數據集（V7-9424）
- 改善CICSUNLOCK支援能力 (V7-9606)
- 處理聯集的預設值大小 (V7-9648)
- JCL/GROOVY處理連接數據集中不同的終止/處置 (V7-9653)
- 為布魯桑資料集進行 pageSize 配置 (V7-9680)
- DSNUTIL-TIME 在 DB2LUW (V7-9697) 中允許載入時有效的 24：00：00
- Support HIGH-在 NumberUtils .ne VALUES ()/ NumberUtils.eq () (V7-9731) 中的 (0XFF) 比較

- JCL/GROOVY-支持做... THENIDCAMSIF THEN-ELSE 子句中的關鍵字，以在單一區塊中執行多個陳述式 (V7-9750)
- 在外部JHDB呼叫的程式無效 JHDBBatchRunner (V7-9782)
- Support SORT OUTFIL 控制卡中的空白字元 (V7-9808)
- 改善CICSREADPREV支援能力 (V7-9845)
- 改善資料集索引的並行存取 (V7-9864)
- 改善CICSREWRITE支援能力 (V7-9873)
- COBOL-支持ACCEPT語句SYSIN中的多行以將數據從 batch（JCL）傳遞到程序（COBOL）（ ）（V7-9875）
- 常規-更好地 ConcatenatedFileConfiguration 處理文件創建步驟（V7-9876）
- IDCAMSUTILITY-處理DEFINEPATH陳述書 (V7-9878)
- SORTBUILD-調整TRAN選項並處理隱含空白 (V7-9925)
- 透CICSDELETE過GENERIC選項支援改善功能 (V7-9939)
- 改善CICSSTARTBR與ENDBR支援 (V7-9952)
- 改善並行存取的關閉效能 (V7-9953)
- 改善啟動時的檔案狀態處理 (V7-9991)
- 常規-允許在getDisposition（V7-10012）上調用getAbnormalTermination（ ）/ ConcatenatedFileConfiguration （ ）getNormalTermination

AS400

新功能

- Support COMMIT 關鍵字的外部指標 (V7-6035)
- SFLCTL寫入後重設讀取迴圈 (V7-8061)
- Support 輕觸式指示燈輸入 CALL (V7-9250)
- 添加新類型的動態字段（拆分）以處理多行上的輸入字段（V7-9370）
- Support 主要/次要檔案 (V7-9390)
- 提交工作時，本機資料區域現在會傳遞至被呼叫的工作 (V7-9775)
- Support QTEMP 資料區域和資料區域值建立的支援。(七九一六)
- 承諾控制-支援啟用/停用承諾控制 (V7-9956)
- Support COMMIT 關鍵字的外部指標

改善

- 改善 0 值顯示功能和 EDTWRD (V7-8933)
- Support DSPF 關鍵字 CHKMSGID "" (V7-9125)
- SQL批次終止時確認交易 (V7-9232)
- 改善對關鍵字EXPORT以及IMPORT欄位和資料結構的支援 (V7-9265)
- 在 DateHelper (V7-9461) 中 Support 小寫字母
- Support CYMD 將 * 轉換為 *ISO（數字）（V7-9488）
- 改善不同欄位的內建 %len 控制點 (運算式的左側和右側) (V7-9733)
- 改善對內建函數「%LOOKUPXX」XX (「LE」、「LT」、「GE」、「GT」) 的支援 (V7-10064)

橫向能力

新功能

- CICS-改善選項狀態的查詢交易（V7-9712）
- JCL-改善系統輸出檔案的系統衝刺負載 (V7-9797)
- CICS-改善 INQUIRE TSQUEUE (V7-9823)
- CICS-改善查詢終端選項使用者識別碼 (V7-9906)

改善

- 改善與空白比較的手柄（V7-8047）
- 改善 JIC 和布魯薩姆的記錄功能 (V7-8847)
- Support 動態欄位的BMS延伸屬性SOSI和程式化符號 F8 (V7-8857)
- 處理程式參數中的緩衝區溢位 (V7-9138)
- 改善布魯桑鎖定登錄的執行緒寫入並行性 (V7-9505)
- Support 公用程式-PGM 的多個資料來源組態 (V7-9570)
- 布魯薩姆記錄級別僅鎖定模式（V7-9626）
- 確保中繼資料持續性能抵抗伺服器重新啟動 (V7-9748)
- 改善例外狀況的DAO清除 (瀏覽器關閉) (V7-9790)
- Sup DummyFile port 對INFUTILBSYSPUNCH象 (V7-9799)
- 增強對於負值的支援 NumericEditedType (V7-9935)

現代化工具 3.6.0 版

主題

- ZoS
- AS400
- 橫向能力

ZoS

新功能

- JCL-加強程序結束時的記錄功能 (V7-8509)
- PL1-提升資料類型的袋子產生能力 PakedLong (V7-8917)
- JCL-當檔案包含「結束」標記時，加強程序結束的記錄功能//(V7-9509)
- PL1-GET EDIT 通過定點和SYSIN流（V7-9593）增強對支持
- DB2-加強對VARGRAPHICDB2類型的支持（V7-9809）
- CICS-改善選項QUERYSECURITY的命令 LOGMESSAGE (V7-9969)
- PL1-改善內置CHARG/字符（V7-9989）的袋子生成

改善

- PL1-加強對INCLUDEX關鍵字的支援 (V7-9588)
- PL/I-處理CHARGRAPHIC關鍵字作為任何方法調用的有效參數（V7-9589）
- 改善使用特定字元 @ # $ § 命名時的PL1主機變數解析度。(第七版)
- COBOL-Support C01... C12 & S01... S05 關鍵字做為剖析步驟中WRITEADVANCING陳述式的參數 (V7-9669)

AS400

新功能

- Sup SQL port-分析儀中的DDS轉換 (V7-7687)
- 自動化 SQL-DDS 檔案偵測 (V7-7687)
- 的實現 SQL-DDS 預處理（V7-7687）

- Sup ALIGN port 關鍵字 (V7-9254)
- Sup ExtName port DSPF 多重尺寸陣列 (V7-9663)
- InvalidKey 關於 COBOL WRITE (V7-9793) 的 Support 聲明

改善

- TESTB對操作碼的改進（V7-8865）
- 改善對焦DECFMT的支援 (V7-8933)
- 處理所產生的指示器 MOVE (V7-9224)
- 改善TEMPLATE對欄位和資料結構關鍵字的支援 (V7-9278)
- 的改進 LIKEDS (使用LIKEDS自動限定的 DS 定義) (V7-9302)
- COBOL-改善指標結構的產生 (V7-9423)
- 原型中的常量參數不是只讀的 (V7-9437)
- 使用編輯代碼「Y」改善EDTCDE關鍵字 (V7-9443)
- Support * ROUTINE 字段的生成PSDS和INFDS（V7-9487）
- 改善重寫欄位XXX為獨立 (重寫時會遺失預設值) (V7-9522)
- 改善對DSPF關鍵字的 Support (V7-9658)
- 處理二進位的ZEROES預設值 (V7-9666)
- Support 隱式指針（V7-9719）
- 使用一個參數改善內建呼叫% 大小的處理 (V7-9730)
- 改善內建呼叫中資料結構參照的處理 (%ELEM) (V7-9736)
- 改善定義規格中具有LIKE參考之欄位簽名長度的處理 (V7-9738)
- 改善項目 REWRITE (V7-9791)
- 從DDS文件生成索引的改進 (V7-9803)
- 使用無效的數值提高映射器的健全性 (V7-9813)
- 改善SQLModel並產生 allIndexes 檔案 (V7-9818)
- 改善合格的 DS 支援 (V7-9863)
- 改善 LOOKUP (使用獨立欄位和參數中LIKE的 DS) (V7-9961) 的支援
- 改善LIKE指標 (V7-9985)
- 處理所產生的指示器 MVR (V7-9995)

- Support 帶有波浪符號的字元 N (V7-10021)
- 改善從SQLDDS舊版DDL檔案產生的現代檔案 (V7-10067)

橫向能力

新功能

- 使用 yml 屬性自訂資源位置 (D88816105)
- COBOL-Support EXIT PERFORM 語句退出內聯PERFORM而不使用 GO TO/PERFORM... THROUGH(V7-9582)
- 指定要納入全域中繼資料的預設舊版編碼。(第七章)

改善

- 改善口罩產生率 (V7-9602)
- 改善上下文預熱功能 (V7-9621)
- 使字符集 CUSTOM93 0 線程安全。(七至九七四)
- 改善項目 MOVEA (V7-9773)

發行版本公告 3.5.0

此版本的 AWS Blu Age 執行階段和現代化工具為 ZoS 和 AS4 00 舊版遷移提供了新功能，主要面向資料集和訊息最佳化，以及擴充的 Java 功能作為轉換程序的結果資產。此版本中的一些關鍵變更包括：

- 除了預先存在的 groovy 腳本功能外，還可以將 CL 程序遷移到 Java 中，以促進其與其他現代化程序的集成，並通過統一生成的編程語言來簡化客戶學習曲線。
- 利用新的資料大量功能，減少 Redis 中資料集載入的時間和最佳化效能。
- 能夠在作業步驟中操作和傳遞資料集，以現代化傳統資料集行為。
- SQL遷移的擴展，以支持 VB 輸入文件和 Java 11 簡化遷移。
- 多種新機制可加快與 IBM MQ 整合的速度，包括額外的標頭、擴充GET/PUT支援，以及佇列中繼資料的自動擷取。
- REST資料集中繼資料的端點，並從 S3 儲存貯體匯入資料集

如需有關此發行版本中所含變更的詳細資訊，請參閱下列各節。

執行階段版本 3.5.0

主題

- ZoS
- AS400
- 橫向能力

ZoS

新功能

- JCLSORT-處理新的關鍵字疊加 (V7-9409)
- ZOSCOBOL-增強對浮動字符的支持（V7-9404）
- 連接埠 RedisJicsTSQueue的 RedisTemplate 連接埠 ListOperations (V7-9212)
- ZOSJCL-如果透過定義，則使用檔案目錄增強暫存目錄的路徑 UserDefinedParameters (V7-9012)
- 處理碼 FUNCTION ORD-MAX 含 ALL (所有陣列項目) (V7-9366)
- 在 Redis 中存儲 TS 隊列時，現在使用前綴和人類可讀的鍵 (V7-9212)
- 為 Blusam 添加獲取數據集端點 API
- JCL-ADD 支持名稱涉及 #（V7-9136）等特殊字符的批處理作業
- TSModel抓取現在被強烈地按需執行（V7-9212）

改善

- 檔案中的非版本化INCLUDE支LNK援 (V7-6022)
- MQ-增強編碼支持（V7-9652）
- 改善對不同字元類型的雙位元組或混合字元集的支援 (V7-9596)
- JCL-Support IDCAMS 刪除NONVSAM陳述式中的 filesDirectory 設定 (V7-9609)
- Support 大量模式ESDS以及從檔案載入RRDS資料集 (V7-8639)
- ESDS在輸入模式下處理空的打開。(七月九 287)
- 使用/ DEFINE CLUSTER 縮UNORD寫ORD支援增強聲明 (V7-9451)
- 布魯桑雷迪斯鎖定性能改進 (V7-8639)
- 增強 DATA () 引數範圍 (V7-9337) 中RECORDSIZE提供的支援DEFINECLUSTER陳述
- 在DEFINECLUSTER語句上添加對BUFFERSPACE/UNIQUE屬性的支持 (V7-9419)

- 改善可變長度記錄資料集的 Blusam 讀取操作。(第一版)
- CICSADDRESS正確地表示缺少CWA為空值 (V7-9491)
- 移除端鎖定處不必要的寫入 (V7-8639)
- 處理快取記憶體中的 Redis 快取範本注入 (V7-9510)
- 正確解碼BPXWDYN參數 (V7-9417)
- 改善出LISTCAT口消耗量 (V7-9201)
- 在布魯桑 TS 隊列名稱中支持不可打印的字符（V7-9212）
- 使用地圖集空處理字段的接收地圖構建 (V7-9486)
- 改進動態訪問模式的 BluesamRelativeFile 刪除和重寫操作。(七九八九)

AS400

新功能

- 添加一個功能，通過標準的 DS/ STM 樞軸生成 CL 文件作為 Java 程序（V7-9427）
- Support 帶ADD模式的輸入文件（V7-9378）
- 改進了排序順序和檢索管理，以支持 cl 命令OPNQRYF（打開查詢文件）並在中添加了對SHARE參數的支持 OverrideItem。(第七章)

改善

- Sup SFLNXTCHG port 於 UpdateSubfile (V7-8061)
- 執行 CL 指令時修改 CL 前後關聯的範圍 (V7-9624)
- 處理程式的傳回碼 BPXWDYN (V7-9417)
- 清除本機監視器。(七至九二四)
- DSPF關鍵字的 S RTNCSRLOC upport (V7-9389)
- setOnGreaterOrEqual() 未將設定為 [等於 1] (V7-9342)
- 更新欄位快取位於 UpdateSubfileRecord (V7-9376)
- 改善 Support 能力 SFLNXTCHG (V7-8061)

橫向能力

新功能

- 忽略文字圖形字串上的 G 前置詞。(七)
- ZOSCOBOL-增強對火災的支持。初始化（ ）對於某些特殊結構（V7-9485）
- 允許以非同步方式初始化內容，以改善程式啟動的效能 (V7-9446)
- SQL明確釋放打開的準備語句和 ResulSet. (七九四二)
- 增強 JMS MQ-支MQRFH2持 MQPUT/V7-7085-支持默認隊列管理器（V7-9400）
- SQL管理-對SET命令的參數啟用 Lambda 轉換 (V7-9492)
- ZOSMQ JMS-將支援新增至MQCOMIT和 MQBACK (V7-9399)
- ZOSIBMMQ-增強對MQINQ（V7-9544）的支持
- 使用雙字節編碼時，使用字節而不是字符串處理CONCAT操作。(七八九 32)
- ZOSIBMMQ-通過選項增強對PUT命令的支持 SET ALL _ _CONTEXT（V7-9544）

改善

- 使用 $ 字符處理 GDG 文件名（V7-9066）
- SQL當前的SQL語NUMBER句成功時，診斷返回 1 作為子句。(第九四十七章)
- 具有非空長度的字段的概述（V7-7536）
- Support 內建PL1GRAPHIC功能 (V7-9245)
- MQ-添加對MQGMO字段設置版本的支持（V7-9500）
- JMSMQ GET-訊息傳回的 dataLength 改善 (V7-9502)
- 設置 sqlerrd (3) 與上下文中獲取的項目的數量。ROWSET(七、九三七一)

現代化工具 3.5.0 版

主題

- ZoS
- AS400
- 橫向能力

ZoS

新功能

- ZOSPLI-Support 使用二進位運算式指派的星號索引 (V7-9178)

- JCL到 BatchScript -一個「//」標誌著作業執行的結束 (V7-9304)
- ZOSPLI-增強對數字編輯類型中浮動字符和簽名的支持（V7-8982）
- COBOL-Support 內建SUM功能 (V7-9367)
- JCL-選擇性地在空白陳述式 (//) 之後註解無效程式碼 (V7-9202)
- JCL-在條件陳述式中 Support 運算子 '|' (V7-9499)
- PL/I-在預處理步驟中對預編譯指令進行註解以防止剖析異常 (V7-9507)

改善

- 使用分隔符處理流定義 (V7-9615)
- 改善LISTCAT出口處理。(七九二○一)
- PL/I-支持隱式「空」參數的增強功能 (V7-9204)

AS400

新功能

- DDS關鍵字的 S CONCAT upport (V7-9439)
- 重構DSPF關鍵字生成的 Java 代碼。(七七七百)
- 在資料結構定義中的欄位上 Support 變異關鍵字 (V7-9029)

改善

- 改善邏輯關係 AND /OR 的剖析 (V7-9352)
- COBOL改善 VO 和之間的映射功能 dsEntity (V7-9449)
- 如果數值輸入被聚焦，則顯示空值 (V7-9374)
- SQL宣告游標中的區域變數 (V7-9456)
- 空白 DS 的範圍問題 (V7-9466)
- 在解析之前截斷 80 列之後的行 (V7-9632)
- 改善定義規範 (V7-9358) 中關鍵字 (DIM、LIKE、...) 中欄位參考和內建呼叫的處理
- Sup SQL port 留言 (-) (V7-9632)
- FullFree 剖析，輸入日期/時間/時間戳記 (V7-9542)
- SQLCA從 FullFree 剖析中包含 (V7-9333)

- 提高控制級別的 Support。(七至九六十)
- 處理器 DS 與 * 的比較 BLANKS (V7-9668)
- 改善對於多個指標的支援 DDS (V7-9318)
- 改善對多個DSPF程式的支援 (V7-9657)
- 改善字段的句柄LIKE（喜歡數據結構的情況和數組中喜歡的數據結構的情況）（V7-9213）
- 免費RPG，在文字上處理延續（V7-9686）
- 改善 Support 程式記錄結束的支援 (V7-9452)
- 語句中的LINKAGE短語的 S CALL upport。(第七版)
- CASXX作業代碼 (CASBB不含CASXX群組) (V7-9357)
- 改善 FullFreeRPG剖析功能 (V7-9457)
- 內建% LEN 不支援 DS 做為引數 (V7-9267)
- MOVEA當因子 2 為 * ALL 'X... '時的改進 (七九二八)
- Support 使用RENAME欄位分配 (V7-9385)

橫向能力

新功能

- SQL移轉工具-在 ebcdic 加載步驟中添加可變記錄長度的OID選項。(七)
- SQL移轉工具-在OID選項上 Support Java 11 (V7-9599)

改善

- 改善對巢狀陣列的支援 (V7-9595)
- 以! 取代字元! 在 '¬ 的情況下，由原始編碼支持。(七九六五)
- JCL-Support PASS 正常終止以在作業步驟之間共用資料集 (V7-9504)
- NULL將 ON 應用於ORACLE何時處理VARCHAR和可為空的數據庫列類型的列定義。(七至九八一)
- 改善彈簧射出合規性 (V7-9635)

AWS 藍光時代的升級說明

本頁包含升級 AWS 藍光時代版本的說明。以下各節提供了從 AWS Blu Age 版本 3.10.0 遷移到 4.0.0 時需要進行的預期更改的完整列表。

從 3 月 10 日移轉至 4.0.0 版

4.0.0 中的主要變化是從春季啟動 2.7 到春季啟動 3.2 和從 Tomcat 9 遷移到 Tomcat 10。

程式碼變更

本節列出了使現代化代碼與 AWS Blu Age 運行時 4.0.0 兼容所需的更改。如果您決定在 Blu Insights (轉型中心) 上使用 4.0.0 版推出新一代產品，則可以略過本節。

POM變化

群組	ArtifactId	變更
組織	俄罗斯阿比	刪除（是傳遞依賴項）
組織羊	蛇怪	刪除（是傳遞依賴項）
組織春季框架. 引導	spring-boot-starter-web	-升級彈簧版本到 3.2.4-刪除日誌 4 4j 的排除 j-to-slf
組織春季框架. 引導	spring-boot-starter-jta-阿托米科斯	更改為可能性：3-起動器：6.0.0 transactions-spring-boot
阿帕切市共享	通用數據庫	升級至 2
組織	Postgreql	升級至 2
微軟. SQL 服務器	姆斯克里特	升級至
阿拉克. 數據庫 .jdbc	相互研究 8	更改至二零零九年三月二十三日版

從賈瓦克斯遷移到雅加達

Tomcat 升級是從 Javax Java 軟件包到雅加達的遷移。確保相應地從 javax.* 更新到雅加達的導入。 *.

Javax 包中幾乎所有舊的引用類都可以在雅加達找到。這種情況的已知例外狀況是`javax.sql`和`javax.xml`套件，它們仍然保持不變。

阿托米克斯改變

由於上述的從屬關係變
更，`org.springframework.boot.jta.atomikos.AtomikosDataSourceBean`必須將的參照變更為`com.atomikos.spring.AtomikosDataSourceBean`。

移除後SQL方言

自訂類別即`PostgreSQLDialect.java`會移除。也必須刪除主啟動器中對其的引用。

部署（AWS 藍色時代運行時（非託管））

Tomcat

此版本與 Tomcat `10.1.17` 兼容。要運行藍光時`4.0.0`代運行時，需要將 Tomcat 服務器升級到此版本。確保移植舊的配置更改（特別是 Catalina 屬性）。

共用相依性

執行階段共用資料夾包含相 up-to-date 依性。

額外的依賴

如果您使用額外的依賴關係（不包含在運行時），則可能需要更新它們。額外資料夾中的 Readme 檔案會列出支援的版本。

AWS 藍色時代運行時概念

了解 AWS Blu 時代運行時的基本概念可以幫助您了解如何通過自動重構實現應用程序現代化。

主題

- AWS 藍色時代運行時高級架構
- AWS 藍光時代結構的現代化應用程序
- AWS 藍光時代的數據簡化器是什麼
- AWS 藍色時代布魯桑
- AWS 藍色時代藍色管理主控台

AWS 藍色時代運行時高級架構

作為將舊版程序現代化為 Java 的 AWS Blu Age 解決方案的一部分， AWS Blu Age Runtime 通過提供傳統構造和程序代碼組織標準化的庫為現代化的應用程序提供了REST基於統一的基於現代化的入口點以及此類應用程序的執行框架。

此類現代化應用程式是 AWS Blu Age 自動化重構程序的結果，用於將大型主機和中型程式（在下文中稱為「舊版」）現代化為基於 Web 的架構。

AWS 藍光時代運行時的目標是傳統程序行為的再現（同功能性），性能（關於程序執行時間和資源消耗），並且易於 Java 開發人員維護現代化程序，儘管使用熟悉的環境和習語，如 tomcat，Spring，geters/setter，流利。APIs

主題

- AWS 藍色時代運行時組件
- 執行環境
- 無狀態和工作階段處理
- 高可用性和無狀態

AWS 藍色時代運行時組件

AWS 藍光時代運行時環境由兩種組件組成：

- 一組 java 庫（jar 文件）通常被引用為「共享文件夾」，並提供遺留的構造和語句。
- 一組 Web 應用程序（戰爭文件）包含基於 Spring 的 Web 應用程序，為現代化程序提供了一組通用的框架和服務。

以下各節將詳細說明這兩個元件的角色。

AWS 藍色時代圖書館

AWS 藍光時代庫是一組存儲在添加到標準 tomcat 類路徑的`shared/`子文件夾中的 jar 文件，以便使它們可用於所有現代化的 Java 程序。他們的目標是提供在 Java 編程環境中既不是本地也不容易使用的功能，而是傳統開發環境的典型功能。這些功能以 Java 開發人員盡可能熟悉的方式公開（geters/setter，基於類的，流利的）。APIs一個重要的例子是數據簡化器庫，它為 Java 程序提供了傳統的內存佈局和操作結構（在PL1或RPG語言中COBOL遇到）。這些罐子是從舊版程序生成的現代化 Java 代碼的核心依賴關係。若要取得有關資料簡化器的更多資訊，請參閱〈〉AWS 藍光時代的數據簡化器是什麼。

Web 應用程式

Web 應用程式歸檔 (WARs) 是將程式碼和應用程式部署到 tomcat 應用程式伺服器的標準方式。作為 AWS Blu Age 運行時的一部分提供的內容旨在提供一組執行框架，重現舊環境和事務監視器（JCL批次CICS，IMS...）以及相關的所需服務。

最重要的一個是gapwalk-application（通常簡稱為「Gapwalk」），它提供了一組統一的REST基於入口點來觸發和控制交易，程序和批次執行。如需詳細資訊，請參閱AWS 藍色時代運行時 APIs。

此 Web 應用程序分配 Java 執行線程和資源，以便在其設計的上下文中運行現代化程序。下節將詳細說明此類複製環境的範例。

其他 Web 應用程序添加到執行環境（更確切地說，在下面描述的「程序註冊表」中）程序，該程序模擬了舊版程序可用並從中調用的程序。這樣的兩個重要類別是：

- 模擬操作系統提供的程序：JCL驅動批次特別希望能夠調用各種文件和數據庫操作程序作為其標準環境的一部分。範例包括SORT/DFSORT或IDCAMS。為此，提供了重現此類行為的 Java 程序，並且可以使用與舊版相同的約定調用。
- 「驅動程序」，這是由執行框架或中間件作為入口點提供的專門程序。一個例子是CBLTDLI，在IMS環境中執行哪些COBOL程序依賴於訪問IMS相關服務（IMS數據庫MFS，用戶對話框等）。

程序註冊表

為了參與並利用這些構造，框架和服務，Java 程序從舊版的現代化遵循. AWS 藍光時代結構的現代化應用程序 在啟動時， AWS Blu Age Runtime 將在一個共同的「程序註冊表」中收集所有這些程序，以便之後可以調用它們（並互相調用）。程序註冊表提供了鬆散的耦合和分解的可能性（因為互相調用的程序不必同時進行現代化）。

執行環境

提供經常遇到的舊式環境和編排：

- JCL驅動批次，一旦現代化為 Java 程序和 Groovy 腳本，可以在同步（阻塞）或異步（分離）方式啟動。在後一種情況下，它們的執行可以通過REST端點進行監視。
- AWS 藍光時代子系統提供類似於以CICS下操作的執行環境：
  - 一個入口點用於啟動CICS事務和運行相關的程序，同時尊重 CICS「運行水平」編排，
  - 資源定義的外部存儲，
  - 一組均勻的 Java 流利APIs複製語句EXEC CICS，
  - 一組可插入的類重現CICS服務，例如臨時存儲隊列，臨時數據隊列或文件訪問（通常可以使用多個實現，例如 Apache Flink 的 Amazon 託管服務，Amazon 簡單隊列服務或 RabbitMQ 用於 TD 隊列），
  - 對於面向用戶的應用程序，BMS屏幕描述格式已現代化為 Angular Web 應用程序，並支持相應的「偽會話」對話框。

- 同樣，另一個子系統提供IMS基於消息的編排，並支持 UI 屏幕的格式現代化。MFS
- 此外，第三個子系統允許在iSeries類似的環境中執行程序，包括DSPF（Display File）指定屏幕的現代化。

所有這些環境都建立在常見的作業系統層級服務之上，例如：

- 傳統內存分配和佈局的仿真（數據簡化器），
- 基於 Java 線程的重現 COBOL「運行單元」執行和參數傳遞機制（CALL語句），
- 模擬平面，串聯，VSAM（通過 Blusam 庫集）和數據集組織，GDG
- 存取資料存放區，例如 RDBMS (EXEC SQL陳述式)。

無狀態和工作階段處理

AWS Blu Age 運行時的一個重要功能是在執行現代化程序時啟用高可用性（HA）和水平可擴展性方案。

這方面的基石是無狀態，其中一個重要的例子是會話處理。HTTP

會話處理

Tomcat 是基於 Web 的，一個重要的機制是HTTP會話處理（由 tomcat 和 Spring 提供）和無狀態設計。由於這樣的無狀態設計是基於以下幾點：

- 用戶雖然連接HTTPS，
- 應用程序服務器部署在負載平衡器後面，
- 當用戶第一次連接到應用程序時，它將進行身份驗證，並且應用程序服務器將創建一個標識符（通常在 cookie 中）
- 此標識符將用作保存和從外部緩存（數據存儲）中檢索用戶上下文的密鑰。

Cookie 管理由 AWS 藍光時代框架和底層 tomcat 服務器自動完成，這對用戶來說是透明的。用戶互聯網瀏覽器將自動管理。

Gapwalk Web 應用程式可能會將工作階段狀態 (內容) 儲存在各種資料存放區中：

- Amazon ElastiCache （雷迪OSS斯）
- 雷迪斯集群
- 在內存映射中（僅適用於開發和獨立環境，不適用於 HA）。

高可用性和無狀態

更一般地說， AWS Blu Age 框架的設計宗旨是無狀態：重現舊版程序行為所需的大多數非暫時狀態不存儲在應用程序服務器中，而是通過外部的通用「單一事實來源」共享。

這些CICS狀態的範例包括「暫存儲隊列」或「資源定義」，而這些狀態的典型外部存儲是與 Redis 兼容的服務器或關聯式數據庫。

這種設計結合負載平衡和共享會話，導致大多數面向用戶的對話框（OLTP「在線交易處理」）在多個「節點」（此處為 tomcat 實例）之間進行分發。

事實上，用戶可以在任何服務器上執行事務，而不關心下一個事務調用是否在不同的服務器上執行。然後，當一個新的服務器產生（由於 auto 擴展，或者替換不健康的服務器），我們可以保證任何可訪問和健康的服務器都可以按預期運行事務並獲得適當的結果（預期返回值，數據庫中的預期數據更改等）。

AWS 藍光時代結構的現代化應用程序

本文件提供現代化應用程式結構的詳細資訊 (使用 AWS 大型主機現代化重構工具)，讓開發人員能夠完成各種工作，例如：

- 流暢地導航到應用程序。
- 開發可以從現代化的應用程序被調用的自定義程序。
- 安全地重構現代化應用程式。

我們假設您已經掌握了以下內容的基本知識：

- 舊式通用編碼概念，例如記錄、資料集及其存取模式，以記錄索引、順序-VSAM、執行單元、jcl 指令碼、CICS概念等。
- java 編碼使用春天框架。
- 在整個文檔中，我們使`short class names`用可讀性。如需詳細資訊，請參AWS 藍色時代完整名稱對應閱擷取 AWS Blu Age 執行階段元素的對應完整名稱，第三方完整名稱對應以及擷取協力廠商元素的對應完整名稱。
- 所有成品和樣本都是從樣本COBOL/CICSCardDemo 應用程序的現代化過程輸出中獲取的。

主題

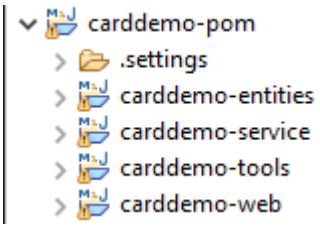
- 成品組織

- 運行和調用程序
- 編寫您自己的程序
- 完整名稱對映

成品組織

AWS 藍光時代現代化的應用程式會封裝成 Java Web 應用程式 (.war)，您可以在伺服器上部署。JEE 通常情況下，服務器是一個嵌入 AWS 藍光時代運行時的 Tomcat 實例，該實例目前建立在 Springboot 和角度（用於 UI 部分）框架上。

戰爭聚合了幾個組件工件（.jar）。每個 jar 都是專用 java 項目的編譯（使用 maven 工具）的結果，該項目的元素是現代化過程的結果。



基本組織依賴於以下結構：

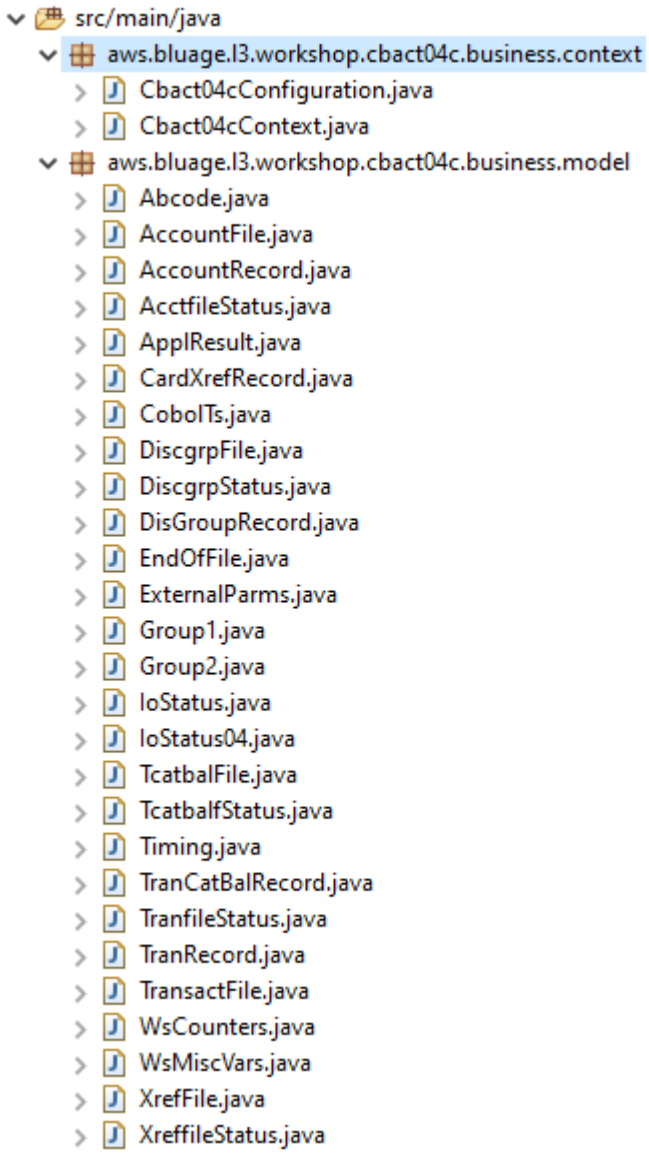
- 實體項目：包含業務模型和上下文元素。項目名稱通常以「-實體」結尾。通常，對於給定的舊版COBOL程序，這對應於 I/O 部分（數據集）和數據分割的現代化。您可以有多個實體專案。
- 服務專案：包含傳統的商務邏輯現代化元素。通常情況下，程序的過程COBOL劃分。您可以擁有多個服務專案。
- 實用程序項目：包含共享的常用工具和實用程序，由其他項目使用。
- Web 項目：包含 UI 相關元素的現代化（如果適用）。不適用於僅批次現代化專案。這些 UI 元素可能來自CICSBMS地圖、IMSMFS元件和其他大型主機 UI 來源。您可以有一個以上的 Web 專案。

實體專案內容

> **Note**
>
> 以下說明僅適用於COBOL和 PL/I 現代化輸出。RPG現代化輸出以不同的配置為基礎。

在進行任何重構之前，實體專案中的封裝組織會繫結至現代化方案。您可以通過幾種不同的方式完成此操作。首選的方法是使用重構工具箱，該工具箱在觸發代碼生成機制之前運行。這是一個高級操作，這

在 BluAge 培訓中進行了說明。如需詳細資訊，請參閱重構研討會。這種方法允許您保留以後重新生成 java 代碼的能力，以便從 future 的進一步改進中受益，例如）。另一種方法是直接在生成的源代碼上進行常規的 java 重構，使用您可能想應用的任何 java 重構方法，風險自擔。



課程相關課程

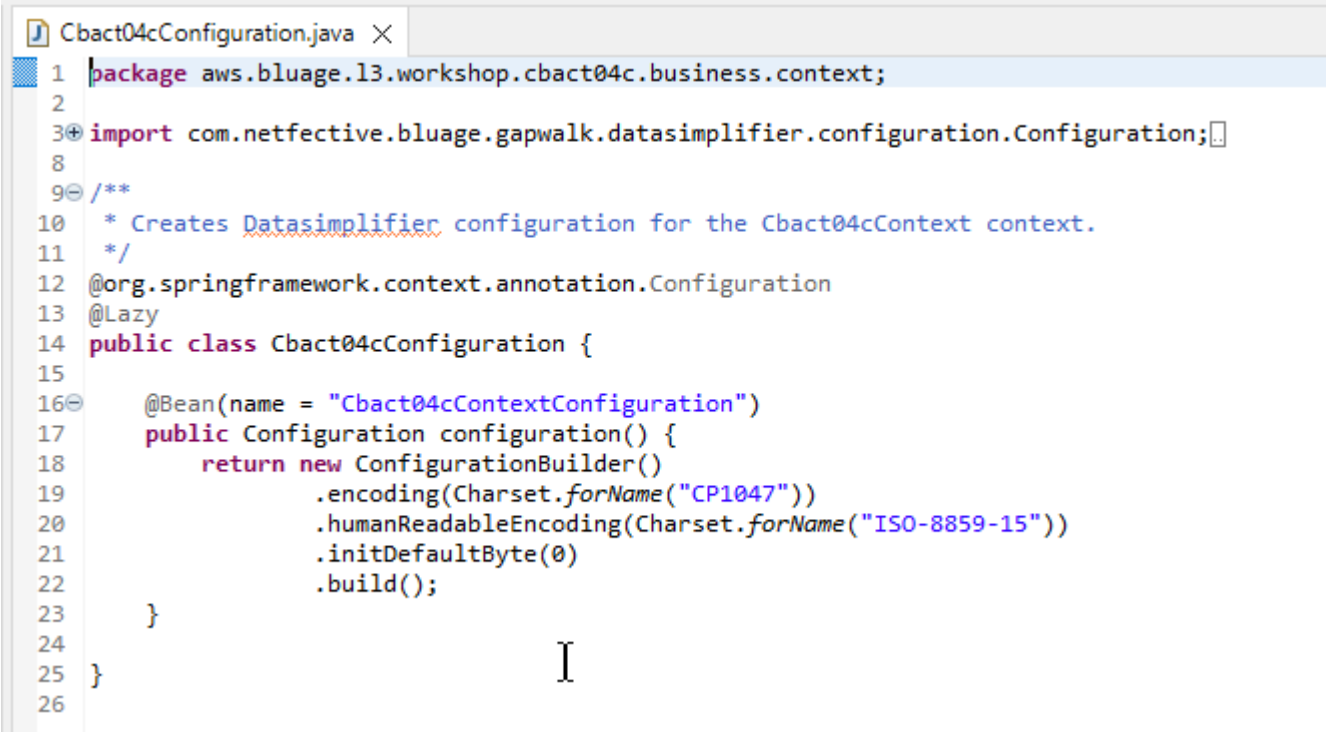
每個現代化程序都與兩個軟件包相關，一個業務 .context 和一個商業 .model 軟件包。

- *base package*.*program*.business.context

  該業務 .context 子包包含兩個類，一個配置類和一個上下文類。

  - 程式的一個組態類別，其中包含指定程式的特定組態詳細資訊，例如用來表示以字元為基礎的資料元素的字元集、填補資料結構元素的預設位元組值等等。類名以「配置」結尾。它

用@org.springframework.context.annotation.Configuration註釋標記，並包含必須返回正確設置Configuration對象的單個方法。



- 一個上下文類，它作為程序服務類（見下文）和數據結構（ ）和數據集（Record）從模型子包（File）之間的橋樑（見下文）。類名以「上下文」結尾，是該類的一個子RuntimeContext類。

Cbact04cContext.java

```
@Component("aws.bluage.l3.workshop.cbact04c.business.context.Cbact04cContext")
@Import({
    aws.bluage.l3.workshop.cbact04c.business.model.TcatbalFile.class
,   aws.bluage.l3.workshop.cbact04c.business.model.XrefFile.class
,   aws.bluage.l3.workshop.cbact04c.business.model.DiscgrpFile.class
,   aws.bluage.l3.workshop.cbact04c.business.model.AccountFile.class
,   aws.bluage.l3.workshop.cbact04c.business.model.TransactFile.class
})
@Lazy
@Scope("prototype")
public class Cbact04cContext extends JicsRuntimeContext {

    @Autowired
    private TcatbalFile tcatbalFile;

    @Autowired
    private XrefFile xrefFile;

    @Autowired
    private DiscgrpFile discgrpFile;

    @Autowired
    private AccountFile accountFile;

    @Autowired
    private TransactFile transactFile;

    private IndexedFile tcatbalFileFile;

    private IndexedFile xrefFileFile;

    private IndexedFile discgrpFileFile;

    private IndexedFile accountFileFile;

    private SequentialFile transactFileFile;

    private TranCatBalRecord tranCatBalRecord;
    private TcatbalfStatus tcatbalfStatus;
    private CardXrefRecord cardXrefRecord;
```

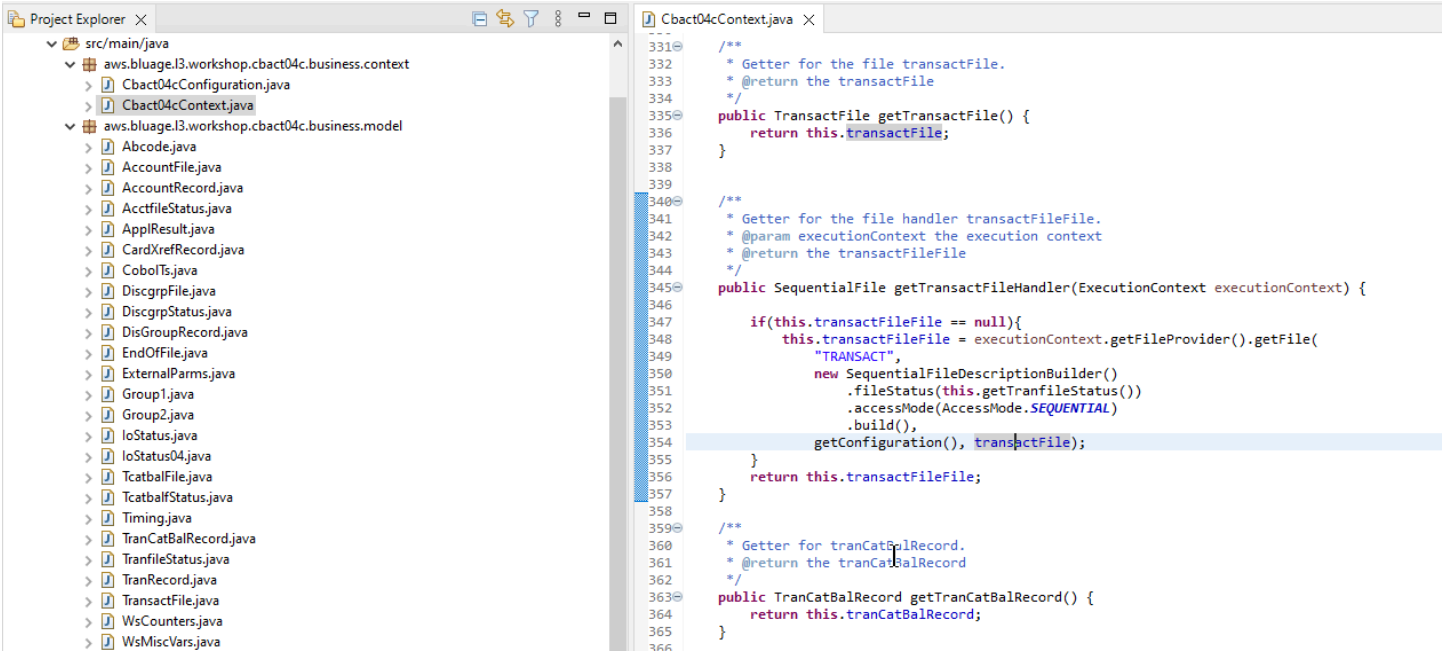
- *base package*.*program*.business.model

  該模型子包包含所有給定的程序可以使用的數據結構。例如，任何 01 級COBOL數據結構對應於模型子包中的一個類（較低級別的數據結構是其擁有 01 級結構的屬性）。有關我們如何現代化 01 數據結構的更多信息，請參閱AWS 藍光時代的數據簡化器是什麼。

DiscgrpFile.java

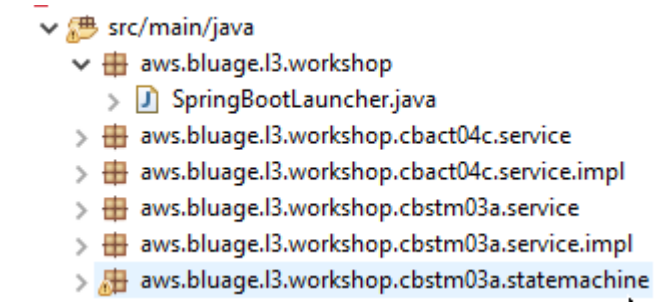
```java
package aws.bluage.l3.workshop.cbact04c.business.model;

import com.netfective.bluage.gapwalk.datasimplifier.configuration.Configuration;
import com.netfective.bluage.gapwalk.datasimplifier.data.structure.Elementary;
import com.netfective.bluage.gapwalk.datasimplifier.data.structure.Group;
import com.netfective.bluage.gapwalk.datasimplifier.entity.ElementaryRangeReference;
import com.netfective.bluage.gapwalk.datasimplifier.entity.RangeReference;
import com.netfective.bluage.gapwalk.datasimplifier.entity.RecordEntity;
import com.netfective.bluage.gapwalk.datasimplifier.metadata.type.AlphanumericType;
import com.netfective.bluage.gapwalk.datasimplifier.metadata.type.ZonedType;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.annotation.Lazy;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

/**
 * Data simplifier file DiscgrpFile.
 *
 * <p>About 'fdDiscgrpRec' field, <br>uml entity: aws.bluage.l3.workshop.cbact04c.business.model.FdDiscgrpRec
 * <br></p>
 *
 */
@Component("aws.bluage.l3.workshop.cbact04c.business.model.DiscgrpFile")
@Lazy
@Scope("prototype")
public class DiscgrpFile extends RecordEntity {

    private final Group root = new Group(getData());
    private final Group fdDiscgrpRec = new Group(root);
    private final Group fdDiscgrpKey = new Group(fdDiscgrpRec);
    private final Elementary fdDisAcctGroupId = new Elementary(fdDiscgrpKey,new AlphanumericType(10));
    private final Elementary fdDisTranTypeCd = new Elementary(fdDiscgrpKey,new AlphanumericType(2));
    private final Elementary fdDisTranCatCd = new Elementary(fdDiscgrpKey,new ZonedType(4, 0, false));
    private final Elementary fdDiscgrpData = new Elementary(fdDiscgrpRec,new AlphanumericType(34));
```

所有類擴展`RecordEntity`類，它代表對業務記錄表示的訪問。一些記錄有一個特殊的目的，因為它們綁定到`File`. a `Record` 和 a 之間的綁定`File`是在創建文件對象時在上下文類中找到的相應 * FileHandler 方法中進行的。例如，下列清單顯示 TransactfileFile `File`如何繫結至 transactFile `Record` (從模型子套件)。



服務項目內容

每個服務項目都配備了一個專用的 Springboot 應用程序，該應用程序被用作架構的骨幹。這是通過位於服務 java 源代碼的基本包中名為SpringBootLauncher的類來實現：



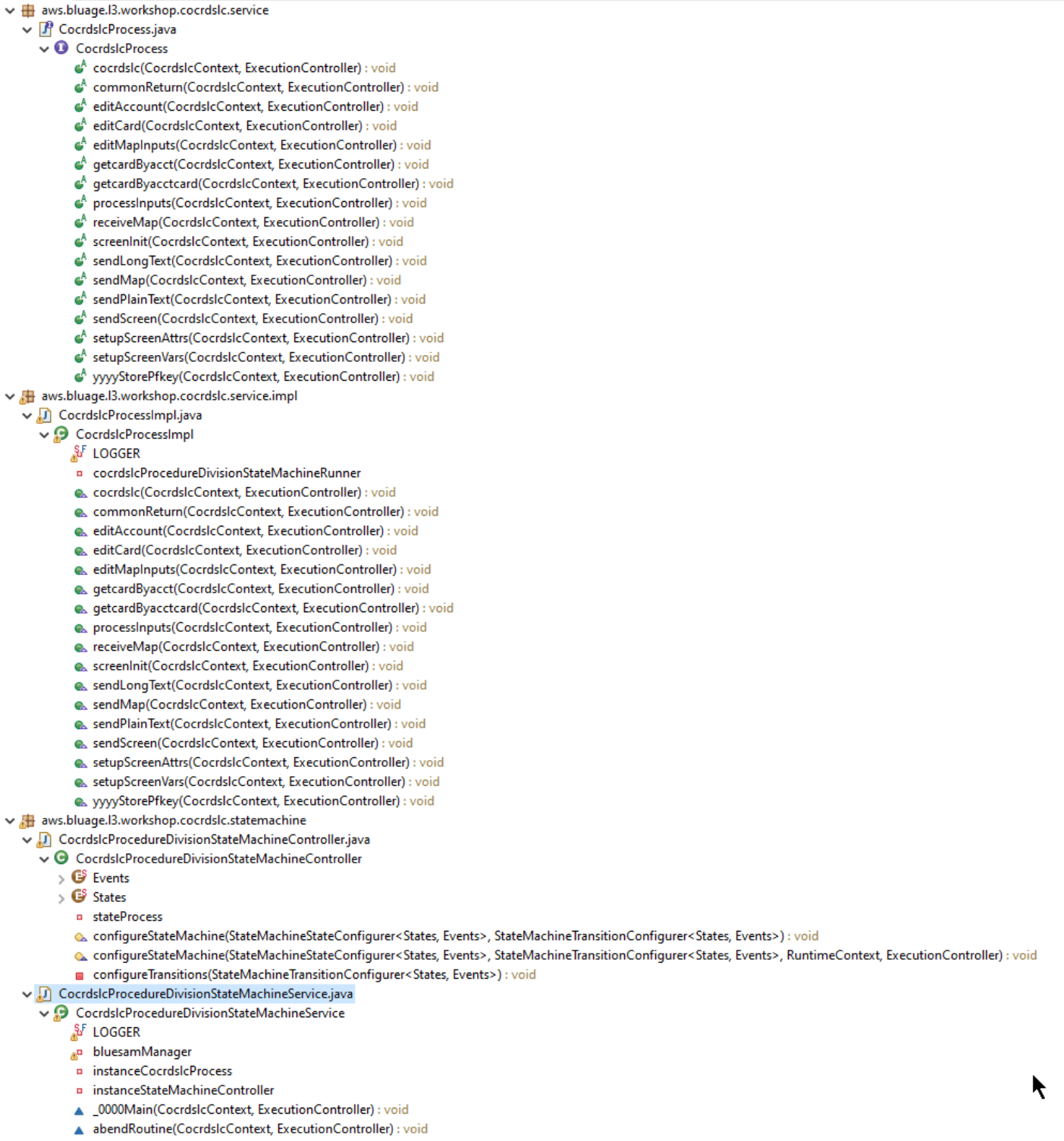
這個課程特別負責：

- 在程序類和託管資源（數據源/事務管理器/數據集映射/等）之間製作膠水。
- 提供一ConfigurableApplicationContext個程序。
- 發現標記為彈簧組件（@Component）的所有類。
- 確保程序正確註冊在 ProgramRegistry--請參閱負責此註冊的初始化方法。

```
/**
 * Initialization method called when the spring application is ready.
 * Register all programs and services to the gapwalk shared context.
 * @param event the application ready event
 */
@EventListener
public void initialize(ApplicationReadyEvent event) {
    Map<String, ProgramContainer> programContainers = event.getApplicationContext().getBeansOfType(ProgramContainer.class);
    programContainers.values().forEach(ProgramRegistry::registerProgram);
    Map<String, ServiceContainer> serviceContainers = event.getApplicationContext().getBeansOfType(ServiceContainer.class);
    serviceContainers.values().forEach(ServiceRegistry::registerService);
}
```

程式相關加工品

在沒有事先重構的情況下，業務邏輯現代化輸出會根據每個舊版程式的兩個或三個套件進行組織：



最詳盡的案例將有三個包：

- *base package*.*program*.service：包含一個名為 Program Pro cess 的介面，該介面具有用於處理業務邏輯的業務方法，並保留傳統執行控制流程。

- *base package*.*program*.service.impl: 包含一個名為 Program 的類ProcessImpl，這是前面描述的 Process 接口的實現。這是遺留語句被「翻譯」為 java 語句的地方，依賴於 AWS Blu Age 框架：

CocrdslcProcessImpl.java

```
210     /**
211      * Process operation sendScreen.
212      *
213      * @param ctx
214      * @param ctrl
215      */
216     @Override
217     public void sendScreen(final CocrdslcContext ctx, final ExecutionController ctrl) {
218         ctx.getCcWorkAreas().setCcardNextMapset(ctx.getWsLiterals().getLitThismapset());
219         ctx.getCcWorkAreas().setCcardNextMap(ctx.getWsLiterals().getLitThismap());
220         ctx.getCarddemoCommarea().setCdemoPgmReenter(true);
221         SendMapBuilder.newInstance(ctx.getDfheiblk(), ctx)
222         .withMap(ctx.getCcWorkAreas().getCcardNextMap())
223         .withMapset(ctx.getCcWorkAreas().getCcardNextMapset())
224         .withData(ctx.getGroup1().getCcrdslaoReference())
225         .withCursor()
226         .withErase()
227         .withFreeKB()
228         .execute();
229         ctx.getWsMiscStorage().setWsRespCd(ctx.getDfheiblk().getEibresp());
230     }
231
232     /**
233      * Process operation processInputs.
234      *
235      * @param ctx
236      * @param ctrl
237      */
238     @Override
239     public void processInputs(final CocrdslcContext ctx, final ExecutionController ctrl) {
240         receiveMap(ctx, ctrl);
241         editMapInputs(ctx, ctrl);
242         ctx.getCcWorkAreas().setCcardErrorMsg(ctx.getWsMiscStorage().getWsReturnMsg());
243         ctx.getCcWorkAreas().setCcardNextProg(ctx.getWsLiterals().getLitThispgm());
244         ctx.getCcWorkAreas().setCcardNextMapset(ctx.getWsLiterals().getLitThismapset());
245         ctx.getCcWorkAreas().setCcardNextMap(ctx.getWsLiterals().getLitThismap());
246     }
247
```

- *base package*.*program*.statemachine：這個軟件包可能並不總是存在。當傳統控制流程的現代化必須使用狀態機器方法（即使用 S pring StateMachine 框架）來正確覆蓋傳統執行流程時，這是必需的。

  在這種情況下，靜態子包包含兩個類：

  - *Program*ProcedureDivisionStateMachineController: 擴充用於驅動 Spring 狀態機力學的類別 StateMachineController (定義控制狀態機執行所需的作業) 和 StateMachineRunner (定義執行狀態機所需的作業) 介面的類別；例如，SimpleStateMachineController如範例中所示。

CocrdslcProcedureDivisionStateMachineController.java

```java
package aws.bluage.l3.workshop.cocrdslc.statemachine;

import aws.bluage.l3.workshop.cocrdslc.business.context.CocrdslcContext;

/**
 * Controller managing the state machine "CocrdslcProcedureDivisionStateMachine" execution.
 */
@Component("aws.bluage.l3.workshop.cocrdslc.statemachine.CocrdslcProcedureDivisionStateMachineController")
@Import({
aws.bluage.l3.workshop.cocrdslc.statemachine.CocrdslcProcedureDivisionStateMachineService.class
})
@Lazy
public class CocrdslcProcedureDivisionStateMachineController extends SimpleStateMachineController<States, Events> {

    /**
     * State machine states.
     */
    public enum States {
        _0000_MAIN_1, _0000_MAIN, ABEND_ROUTINE, FINAL, LOCAL_FINAL
    }

    /**
     * State machine events.
     */
    public enum Events {
        TO__0000_MAIN_1, TO__0000_MAIN, TO_ABEND_ROUTINE, TO_FINAL, TO_LOCAL_FINAL
    }

    /**
     * State machine state process service provider.
     */
    @Autowired
    @Lazy
    private CocrdslcProcedureDivisionStateMachineService stateProcess;

    @Override
    protected void configureStateMachine(StateMachineStateConfigurer<States, Events> states, StateMachineTransitionConfigurer<States, Events> transitions) throws Exception {
        throw new UnsupportedOperationException("Please use the four arguments configureStateMachine method instead: configureStateMachine(StateMachineStateConfigurer<States, Events> states, "
                + "StateMachineTransitionConfigurer<States, Events> transitions, RuntimeContext ctx, ExecutionController ctrl)");
    }

    @Override
    protected void configureStateMachine(StateMachineStateConfigurer<States, Events> states, StateMachineTransitionConfigurer<States, Events> transitions, RuntimeContext ctx, ExecutionController ctrl) throws Exception {
        StateConfigurer<States, Events> configurer = states.withStates();
        configurer.initial(States._0000_MAIN_1).end(States.FINAL);
        configurer.state(States._0000_MAIN_1);
        configurer.state(States.FINAL);

        StateConfigurer<States, Events> subConfigurer = states.withStates().parent(States._0000_MAIN_1);
        subConfigurer.initial(States._0000_MAIN).end(States.LOCAL_FINAL);
        CocrdslcContext lctx = (CocrdslcContext) ctx;
        subConfigurer.state(States._0000_MAIN, buildAction(() -> {stateProcess._0000Main(lctx, ctrl);}), null);
        subConfigurer.state(States.ABEND_ROUTINE, buildAction(() -> {stateProcess.abendRoutine(lctx, ctrl);}), null);

        configureTransitions(transitions);
    }

    /**
     * Declare state machine transitions.
     * @param transitions the transitions configuration helper
     */
    private void configureTransitions(StateMachineTransitionConfigurer<States, Events> transitions) throws Exception {
        transitions.withLocal().source(States._0000_MAIN_1).target(States.ABEND_ROUTINE).event(Events.TO_ABEND_ROUTINE);
        transitions.withExternal().source(States.ABEND_ROUTINE).target(States.FINAL).event(Events.TO_FINAL);
    }
}

```

狀態機控制器定義了可能的不同狀態和它們之間的轉換，從而重現給定程序的傳統執行控制流程。

在構建狀態機時，控制器指的是在位於狀態機包中的相關服務類中定義的方法，如下所述：

```
subConfigurer.state(States._0000_MAIN, buildAction(() ->
 {stateProcess._0000Main(lctx, ctrl);}), null);
subConfigurer.state(States.ABEND_ROUTINE, buildAction(() ->
 {stateProcess.abendRoutine(lctx, ctrl);}), null);
```

- *Program*ProcedureDivisionStateMachineService: 此服務類別代表一些需要與狀態機控制器建立之狀態機器繫結的業務邏輯，如前所述。

  在這個類的方法的代碼使用狀態機控制器中定義的事件：

CocrdslcProcedureDivisionStateMachineService.java

```
59      /**
60       * State process operation _0000Main.
61       *
62       * @param ctx
63       * @param ctrl
64       */
65      void _0000Main(CocrdslcContext ctx, ExecutionController ctrl) {
66          ctx.getDfheiblk().bind(ArgUtils.get(ctx, 0));
67          ctx.getDfhcommarea().bind(ArgUtils.get(ctx, 1));
68
69          /*
70          *****************************************************************
71          Program:     COCRDSLC.CBL                                      *
72          Layer:       Business logic                                    *
73          Function:    Accept and process credit card detail request     *
74          *****************************************************************
75          Copyright Amazon.com, Inc. or its affiliates.
76          All Rights Reserved.
77          Licensed under the Apache License, Version 2.0 (the \"License\").
78          You may not use this file except in compliance with the License.
79          You may obtain a copy of the License at
80          http://www.apache.org/licenses/LICENSE-2.0
81          Unless required by applicable law or agreed to in writing,
82          software distributed under the License is distributed on an
83          \"AS IS\" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
84          either express or implied. See the License for the specific
85          language governing permissions and limitations under the License
86          *****************************************************************
87          Ver: CardDemo_v1.0-15-g27d6c6f-68 Date: 2022-07-19 23:16:00 CDT */
88          instanceStateMachineController.registerSignalHandler(Events.TO_ABEND_ROUTINE, "!ABEND");
89          HandleAbendBuilder.newInstance(ctx.getDfheiblk(), ctx).execute().handleException();
90          ctx.getCcWorkAreas().getCcWorkAreaReference().getField().initialize();
91          ctx.getWsMiscStorage().getField().initialize();
92          DataUtils.initialize(ctx.getWsCommarea().getWsCommareaReference());
93
```

CocrdslcProcedureDivisionStateMachineService.java

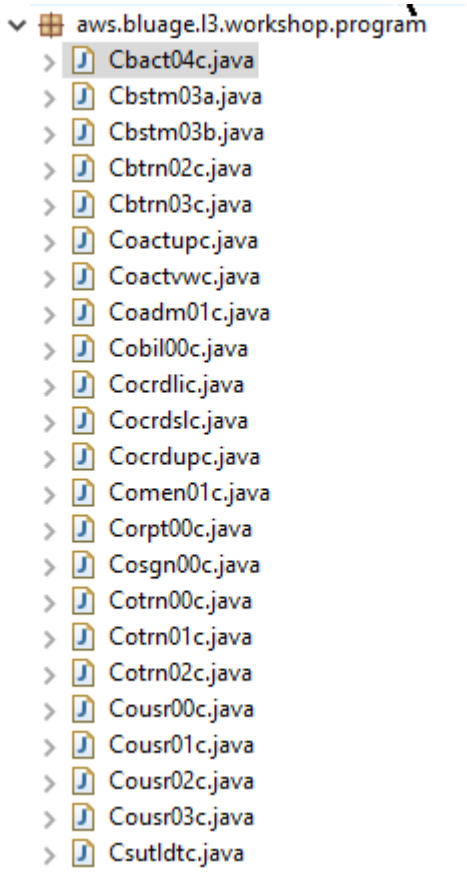
```
221     *
222     * @param ctx
223     * @param ctrl
224     */
225    void abendRoutine(CocrdslcContext ctx, ExecutionController ctrl) {
226        if (DataUtils.isLowValue(ctx.getAbendData().getAbendMsgReference())) {
227            ctx.getAbendData().setAbendMsg("UNEXPECTED ABEND OCCURRED.");
228        }
229        ctx.getAbendData().setAbendCulprit(ctx.getWsLiterals().getLitThispgm());
230        SendTextBuilder.newInstance(ctx.getDfheiblk(), ctx)
231        .withData(ctx.getAbendData())
232        .withLength(134)
233        .execute();
234        HandleAbendBuilder.newInstance(ctx.getDfheiblk(), ctx).cancel().execute().handleException();
235        AbendBuilder.newInstance(ctx.getDfheiblk(), ctx).withAbendCode("9999").execute().handleException();
236
237        /*
238        Ver: CardDemo_v1.0-15-g27d6c6f-68 Date: 2022-07-19 23:12:33 CDT */
239        instanceStateMachineController.sendEvent(Events.TO_FINAL);
240
241    }
242 }
243
```

靜態服務也會呼叫先前描述的程序服務實作：

CocrdslcProcedureDivisionStateMachineService.java

```
166
167             /*
168             ************************************************************
169             COMING FROM CREDIT CARD LIST SCREEN
170             SELECTION CRITERIA ALREADY VALIDATED
171             ************************************************************ */
172         } else if (ctx.getCarddemoCommarea().isCdemoPgmEnter() && DataUtils.compare(ctx.getCarddemoCommarea().getCdemoFromProgramReference(), ctx.getWsLiterals().getLitCclistpgmReference()) == 0) {
173             ctx.getWsMiscStorage().setInputOk(true);
174             ctx.getCcWorkAreas().setCcAcctIdN(ctx.getCarddemoCommarea().getCdemoAcctId());
175             ctx.getCcWorkAreas().setCcCardNumN(ctx.getCarddemoCommarea().getCdemoCardNum());
176             instanceCocrdslcProcess.getcardByacctcard(ctx, ctrl);
177             instanceCocrdslcProcess.sendMap(ctx, ctrl);
178             instanceCocrdslcProcess.commonReturn(ctx, ctrl);
179         } else if (ctx.getCarddemoCommarea().isCdemoPgmEnter()) {
180
181             /*
182             ************************************************************
183             COMING FROM SOME OTHER CONTEXT
184             SELECTION CRITERIA TO BE GATHERED
185             ************************************************************ */
186             instanceCocrdslcProcess.sendMap(ctx, ctrl);
187             instanceCocrdslcProcess.commonReturn(ctx, ctrl);
188         } else if (ctx.getCarddemoCommarea().isCdemoPgmReenter()) {
189             instanceCocrdslcProcess.processInputs(ctx, ctrl);
190             if (ctx.getWsMiscStorage().isInputError()) {
191                 instanceCocrdslcProcess.sendMap(ctx, ctrl);
192                 instanceCocrdslcProcess.commonReturn(ctx, ctrl);
193             } else {
194                 instanceCocrdslcProcess.getcardByacctcard(ctx, ctrl);
195                 instanceCocrdslcProcess.sendMap(ctx, ctrl);
196                 instanceCocrdslcProcess.commonReturn(ctx, ctrl);
197             }
198         } else {
199             ctx.getAbendData().setAbendCulprit(ctx.getWsLiterals().getLitThispgm());
200             ctx.getAbendData().setAbendCode("0001");
201             DataUtils.setToBlank(ctx.getAbendData().getAbendReasonReference());
202             ctx.getWsMiscStorage().setWsReturnMsg("UNEXPECTED DATA SCENARIO");
203             instanceCocrdslcProcess.sendPlainText(ctx, ctrl);
204         }
205
206         /*
207         If we had an error setup error message that slipped through
208         Display and return */
209         if (ctx.getWsMiscStorage().isInputError()) {
210             ctx.getCcWorkAreas().setCcardErrorMsg(ctx.getWsMiscStorage().getWsReturnMsg());
211             instanceCocrdslcProcess.sendMap(ctx, ctrl);
212             instanceCocrdslcProcess.commonReturn(ctx, ctrl);
213         }
214         instanceCocrdslcProcess.commonReturn(ctx, ctrl);
```

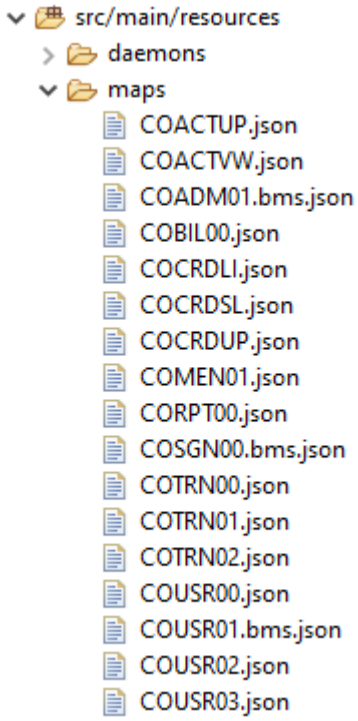
除此之外，一個名為的軟件包*base package*.program起著重要作用，因為它每個程序收集一個類，這將作為程序進入點（稍後有關此更多詳細信息）。每個類實現接Program口，標記為程序入口點。



其他文物

- BMSMAPs同伴

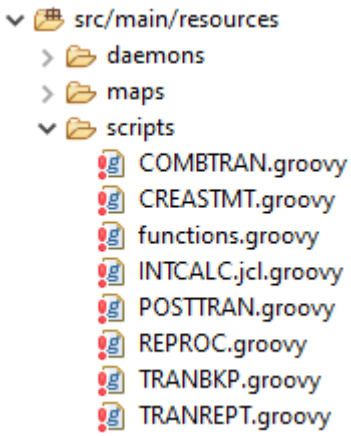
  除了方案相關的人工因素之外，服務專案還可以包含用於各種目的的的其他成品。在CICS線上應用程式的現代化情況下，現代化程序會產生一個 json 檔案，並放入 /src/main/resources 資料夾的地圖資料夾中：



Blu Age 運行時消耗這些 json 文件，以將SENDMAP語句使用的記錄與屏幕字段綁定。

- 常規的腳本

  如果傳統應用程序有JCL腳本，那麼這些腳本已經被現代化為 groovy 腳本，存儲在 /src/main/資源/腳本文件夾中（稍後有關該特定位置的更多信息）：

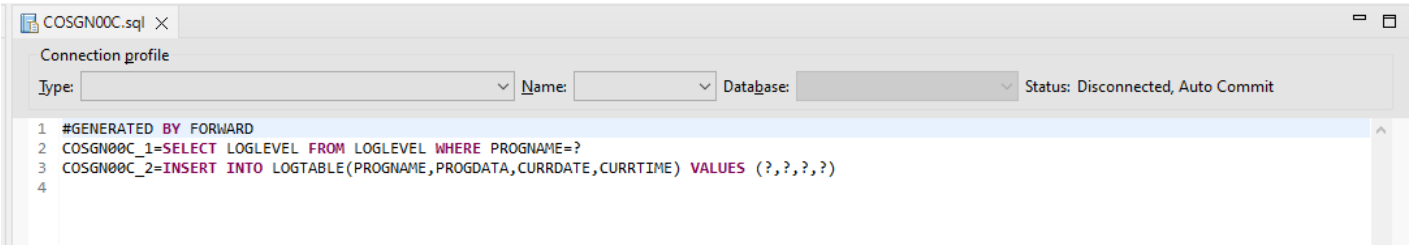


  這些指令碼可用來啟動批次工作 (專用、非互動式、CPU 密集型資料處理工作負載)。

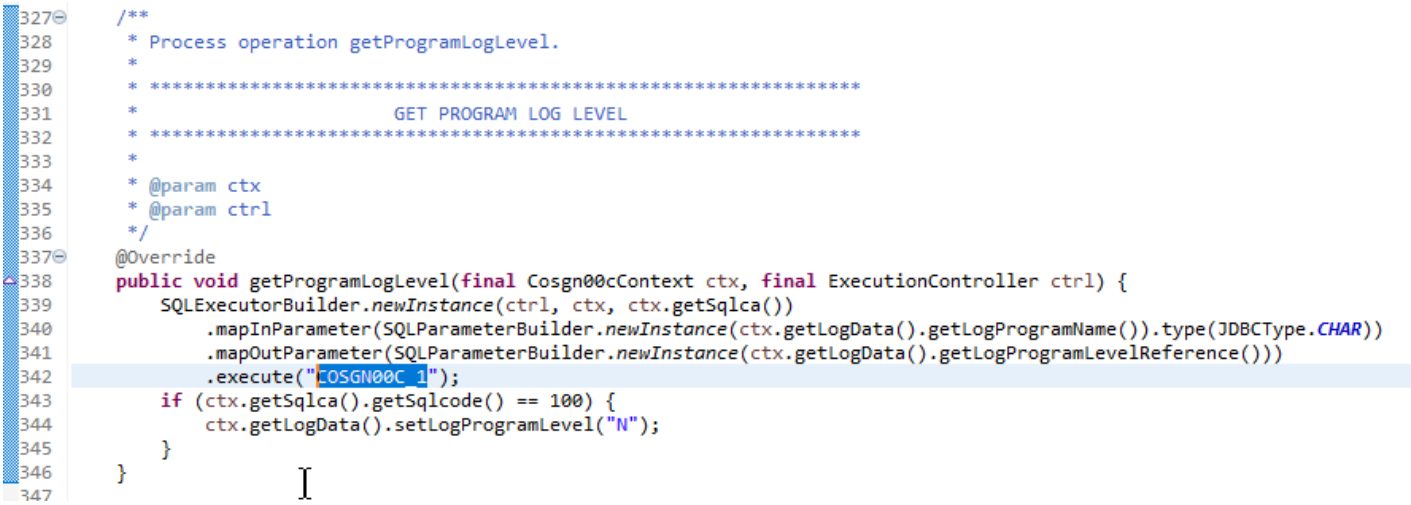
- SQL文件

  如果舊版應用程式使用SQL查詢，則相應的現代化SQL查詢已收集到專用屬性檔案中，並使用命名模式程式 .sql，其中的程式是使用這些查詢的程式名稱。

這些 sql 文件的內容是（key=query）條目的集合，其中每個查詢都與一個唯一密鑰相關聯，現代化程序用於運行給定的查詢：

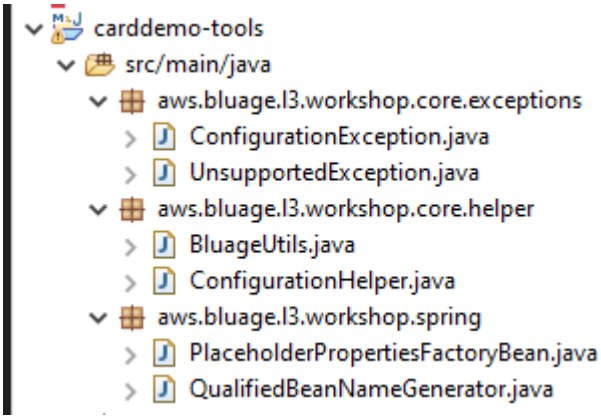


例如，COSGN00C 程序正在使用密鑰「COSGN00C_1」（sql 文件中的第一個條目）執行查詢：



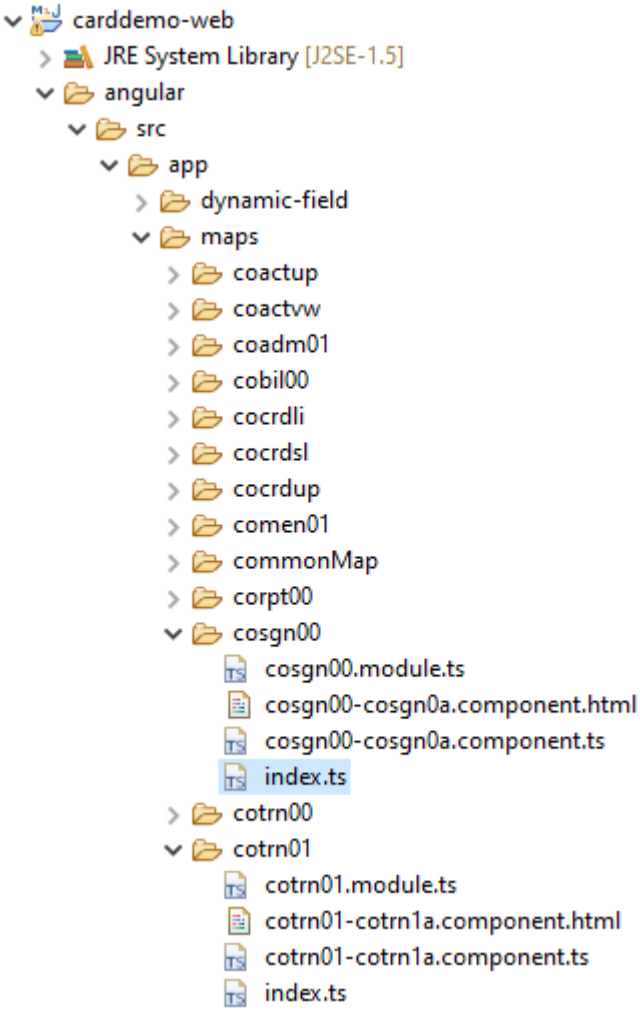
公用程式專案內

公用事業項目，其名稱以「-tools」結尾，包含一組技術實用程序，可能被所有其他項目使用。



Web 專案內容

Web 項目只有在現代化傳統 UI 元素時才存在。用於構建現代化的應用程序前端的現代 UI 元素基於 Angular。用來顯示現代化成品的範例應用程式是在大型主機上執行的COBOL/CICS應用程式。系CICS統使用MAPs來表示 UI 螢幕。相應的現代元素將是，對於每個地圖，一個 html 文件伴隨著打字稿文件：



Web 專案只負責應用程式的前端方面依賴公用程式和實體專案的服務專案提供後端服務。前端和後端之間的鏈接是通過名為 Gapwalk-應用程序，這是標準的 AWS 藍光時代運行時分佈的一部分的 Web 應用程序進行。

運行和調用程序

在傳統系統上，程式會編譯為獨立的可執行檔，可透過CALL機制 (例如COBOLCALL陳述式) 自行呼叫，並在需要時傳遞引數。現代化的應用程式提供相同的功能，但使用不同的方法，因為所涉及的成品的性質與舊版的性質不同。

在現代化方面，程序入口點是實現`Program`接口的特定類，是 Spring 組件（@Component），並且位於服務項目中，名為*`base package`*`.program`。

程序註冊

每次託管現代化應用程序的 Tomcat 服務器啟動時，服務 Springboot 應用程序也被啟動，這會觸發程序註冊。一個名`ProgramRegistry`為專用的註冊表填充了程序條目，每個程序正在使用它的標識符，每個已知的程序標識符一個條目，這意味著，如果一個程序是由幾個不同的標識符已知，註冊表包含盡可能多的條目，因為有標識符。

給定程序的註冊依賴於由 getProgramIdentifiers（）方法返回的標識符的集合：

Cbact04c.java

```java
package aws.bluage.l3.workshop.program;

import aws.bluage.l3.workshop.SpringBootLauncher;

/**
 * Reference the spring application of program CBACT04C.
 *  Provides an access to the contained program for the run unit.
 */
@Component
@Import({
    aws.bluage.l3.workshop.cbact04c.business.context.Cbact04cConfiguration.class,
    aws.bluage.l3.workshop.cbact04c.business.context.Cbact04cContext.class,
    aws.bluage.l3.workshop.cbact04c.service.impl.Cbact04cProcessImpl.class
})
public class Cbact04c implements Program {
    /**
     * Unique identifiers for the contained program.
     */
    private static final Set<String> programIdentifiers = Collections.unmodifiableSet(Stream.of("CBACT04C").collect(Collectors.toSet()));

    /**
     * Main program identifier for the contained program.
     */
    private static final String programIdentifier = "CBACT04C";
    @Autowired
    PlatformTransactionManager transactionManager;

    @Autowired
    Map<String, DataSource> datasources;
    @Autowired
    BeanFactory beanFactory;
    /**
     * {@inheritDoc}
     */
    @Override
    public ConfigurableApplicationContext getSpringApplication() {
        return SpringBootLauncher.getCac();
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public void updateExecutionContext(ExecutionContext executionContext) {
        executionContext.setDatasources(datasources);
        executionContext.setDatabaseSupport(ExecutionContext.DatabaseSupport.POSTGRE);
        executionContext.setSqlcaVersion(ExecutionContext.SqlcaVersion.getEnum("ansi-comp5"));
        executionContext.setTransactionManager(transactionManager);
        executionContext.setUseSQLDateNewParadigm(true);
        executionContext.setUseSQLTrimStringType(false);
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public Set<String> getProgramIdentifiers() {
        return programIdentifiers;
    }
```

在這個例子中，該程序被註冊一次，名為 'CBACT04C'（看看 programIdentifiers 集合的內容）。tomcat 日誌顯示每個程序註冊。程序註冊僅取決於聲明的程序標識符，而不是程序類名本身（儘管通常程序標識符和程序類名稱是對齊的。

相同的註冊機制適用於各種實用程序 AWS Blu Age Web 應用程序帶來的實用程序，這些應用程序是 AWS Blu Age 運行時分佈的一部分。例如，Gapwalk-Utiliy-Pgm web 應用程式提供 z/OS 系統公用程式 (IDCAMS、、等等) 的功能對等項目，ICEGENER而且可以透過現代化的程式或指令碼來呼叫。SORT在 Tomcat 啟動時註冊的所有可用公用程式都會記錄在 Tomcat 記錄檔中。

腳本和守護進程註冊

類似的註冊過程，在 Tomcat 啟動時，發生在/src /主/資源/腳本文件夾層次結構的groovy 腳本。腳本文件夾層次結構被遍歷，並且發現的所有 groovy 腳本（除了特殊的 functions.groovy 保留腳本）都在中註冊`ScriptRegistry`，並使用它們的短名稱（腳本文件名的一部分位於第一個點字符之前）作為檢索的關鍵字。

> **Note**
>
> - 如果有多個腳本具有將導致產生相同註冊密鑰的文件名，則只會註冊最新的腳本，覆蓋任何先前遇到的該給定密鑰的註冊。
> - 考慮到上述注意事項，使用子文件夾時請注意，因為註冊機制會扁平化層次結構並可能導致意外的覆蓋。層次結構在註冊過程中不計算：典型地/腳本/A /MyScript.Groovy 和/腳本 /B/MyScript.Groovy 將導致/腳本//MySCRIPT。

/src/主/資源/守程文件夾中的groovy 腳本的處理方式有點不同。它們仍然註冊為常規腳本，但此外，它們會直接在 Tomcat 啟動時間以異步方式啟動一次。

在中註冊指令碼之後`ScriptRegistry`，REST呼叫可以使用 Gapwalk 應用程式公開的專用端點來啟動它們。如需詳細資訊，請參閱對應的文件。

程序調用程序

每個程序可以調用另一個程序作為一個子程序，傳遞參數給它。程序使用接`ExecutionController`口的實現來這樣做（大多數情況下，這將是一個實`ExecutionControllerImpl`例），以及一個名為 the 的流暢API機制`CallBuilder`來構建程序調用參數。

所有程序方法都採用 a `RuntimeContext` 和 a `ExecutionController` as 方法參數，因此始終可用於調用其他程序。`ExecutionController`

例如，請參閱下面的圖表，其中顯示 03 CBST A 程序如何調用 CBST 03B 程序作為子程序，並將參數傳遞給它：

CbstmO3aProcessImpl.java

```java
    /**
     * Process operation xreffileGetNext.
     *
     * ----------------------------------------------------------------*
     *
     * @param ctx
     * @param ctrl
     */
    @Override
    public void xreffileGetNext(final Cbstm03aContext ctx, final ExecutionController ctrl) {
        ctx.getWsM03bArea().setWsM03bDd("XREFFILE");
        ctx.getWsM03bArea().setM03bRead(true);
        DataUtils.setToZeroes(ctx.getWsM03bArea().getWsM03bRcReference());
        DataUtils.setToBlank(ctx.getWsM03bArea().getWsM03bFldtReference());
        ctrl.callSubProgram("CBSTM03B", CallBuilder.newInstance()
            .byReference(ctx.getWsM03bArea())
            .getArguments(), ctx);
        if (DataUtils.compare(ctx.getWsM03bArea().getWsM03bRcReference(), "00") == 0) {

            /*
            Do nothing */
        } else if (DataUtils.compare(ctx.getWsM03bArea().getWsM03bRcReference(), "10") == 0) {
            ctx.getMiscVariables().setEndOfFile("Y");
        } else {
            if (LOGGER.isInfoEnabled()) LOGGER.info("ERROR READING XREFFILE");
            if (LOGGER.isInfoEnabled()) LOGGER.info("{}{}" , "RETURN CODE: " , ctx.getWsM03bArea().getWsM03bRc());
            abendProgram(ctx, ctrl);
        }
        ctx.getCardXrefRecord().setBytes(ctx.getWsM03bArea().getWsM03bFldtReference().getBytes());
    }
```

- 的第一個引數`ExecutionController.callSubProgram`是要調用的程序的標識符（即用於程序註冊的標識符之一--見上面的段落）。
- 第二個引數，這是構建的結果`CallBuilder`，是一個數組`Record`，對應於從調用者傳遞給被調用者的數據。
- 第三個也是最後一個引數是呼叫者`RuntimeContext`實例。

所有這三個參數都是強制性的，不能為 null，但第二個參數可以是一個空數組。

被調用者將能夠處理傳遞的參數，只有當它最初被設計為這樣做。對於傳統程COBOL序，這意味著具有一個LINKAGE部分和一個USING子句，用於程序劃分以利用這些LINKAGE元素。

例如，請參閱相應的 CBSTM03B。 CBLCOBOL來源檔案：

github.com/aws-samples/aws-mainframe-modernization-carddemo/blob/main/app/cbl/CBSTM03B.CBL

```
 98
 99          LINKAGE SECTION.
100          01  LK-M03B-AREA.
101              05  LK-M03B-DD          PIC X(08).
102              05  LK-M03B-OPER        PIC X(01).
103                88  M03B-OPEN       VALUE 'O'.
104                88  M03B-CLOSE      VALUE 'C'.
105                88  M03B-READ       VALUE 'R'.
106                88  M03B-READ-K     VALUE 'K'.
107                88  M03B-WRITE      VALUE 'W'.
108                88  M03B-REWRITE    VALUE 'Z'.
109              05  LK-M03B-RC          PIC X(02).
110              05  LK-M03B-KEY         PIC X(25).
111              05  LK-M03B-KEY-LN      PIC S9(4).
112              05  LK-M03B-FLDT        PIC X(1000).
113
114          PROCEDURE DIVISION USING LK-M03B-AREA.
115
```

所以 CBSTM 03B 程序將一個單個Record作為參數（大小為 1 的數組）。這是使用byReference（）和getArguments（）方法鏈接的構建。CallBuilder

CallBuilder流利的API類有幾種方法可用於填充參數數組以傳遞給被調用者：

- asPointer（RecordAdaptable）：通過引用添加指針類型的參數。指針表示目標數據結構的地址。
- byReference（RecordAdaptable）：通過引用添加參數。呼叫者將看到被呼叫者執行的修改。
- byReference（RecordAdaptable）：以前方法的可變參數變體。
- byValue（對象）：添加一個參數Record，按值轉換為 a。呼叫者不會看到被呼叫者執行的修改。
- byValue（RecordAdaptable）：與前面的方法相同，但參數直接作為RecordAdaptable.
- byValueWith邊界（Object，int，int）：添加一個參數，轉換為 aRecord，按值提取由給定邊界定義的字節數組部分。

最後，該 getArguments 方法將收集所有添加的參數並將其作為數組返回Record。

> **Note**
>
> 確保 arguments 數組具有所需的大小，就內存佈局與鏈接元素的預期佈局而言，這些項目是正確排序和兼容的調用者的響應性。

腳本調用程序

從 groovy 腳本調用註冊的程序需要使用實現`MainProgramRunner`接口的類實例。通常，獲得這樣的實例是通過 Spring 的 ApplicationContext 使用來實現的：

REPROC.groovy

```groovy
// Import
import com.netfective.bluage.gapwalk.rt.provider.ScriptRegistry
import com.netfective.bluage.gapwalk.rt.call.MainProgramRunner
import com.netfective.bluage.gapwalk.io.support.FileConfigurationUtils
import com.netfective.bluage.gapwalk.rt.job.support.DefaultJobContext
import com.netfective.bluage.gapwalk.rt.utils.GroovyUtils
import com.netfective.bluage.gapwalk.rt.io.support.FileConfiguration
import com.netfective.bluage.gapwalk.rt.shared.AbendException
import com.netfective.bluage.gapwalk.rt.call.exception.GroovyExecutionException
// Variables
mpr = applicationContext.getBean("com.netfective.bluage.gapwalk.rt.call.ExecutionController", MainProgramRunner.class)
TreeMap mapTransfo = [:]
```

`MainProgramRunner`接口可用後，使用該 runProgram 方法調用程序並將目標程序的標識符作為參數傳遞：

REPROC.groovy

```
//*******************************************************************
//*                              STEPS                              *
//*******************************************************************
// STEP PRC001 - PGM - IDCAMS*****************************************
def stepPRC001(Object shell, Map params, Map programResults){
    shell.with {
        if (checkValidProgramResults(programResults)) {
            return execStep("PRC001", "IDCAMS", programResults, {
                mpr
                    .withFileConfigurations(new FileConfigurationUtils()
                        .systemOut("SYSPRINT")
                        .output("*")
                        .build()
                        .bluesam("FILEIN")
                        .dataset("NULLFILE")
                        .disposition("SHR")
                        .build()
                        .bluesam("FILEOUT")
                        .dataset("NULLFILE")
                        .disposition("SHR")
                        .build()
                        .fileSystem("SYSIN")
                        .path("&CNTLLIB(REPROCT)")
                        .disposition("SHR")
                        .build()
                        .getFileConfigurations(fcmap))
                    .withParameters(params)
                    .runProgram("IDCAMS")
                })
        }
    }
}
```

在前面的範例中，作業步驟呼叫 IDCAMS (檔案處理公用程式)，提供實際資料集定義及其邏輯識別碼之間的對應。

在處理數據集時，舊版程序大多使用邏輯名稱來識別數據集。從指令碼呼叫程式時，指令碼必須將邏輯名稱與實際的實體資料集對應。這些數據集可以位於文件系統中，在 Blusam 存儲中，甚至由內聯流，多個數據集的串聯或. GDG

使用此方 withFileConfiguration 法建立資料集的邏輯對實體對應，並將其提供給被呼叫的程式。

編寫您自己的程序

為腳本或其他現代化程序編寫自己的程序來調用是一項常見任務。一般而言，在現代化專案中，當可執行的舊版程式是以現代化程序不支援的語言撰寫，或是來源已遺失 (是的，可能發生這種情況)，或程式是無法取得來源的公用程式時，撰寫自己的程式。

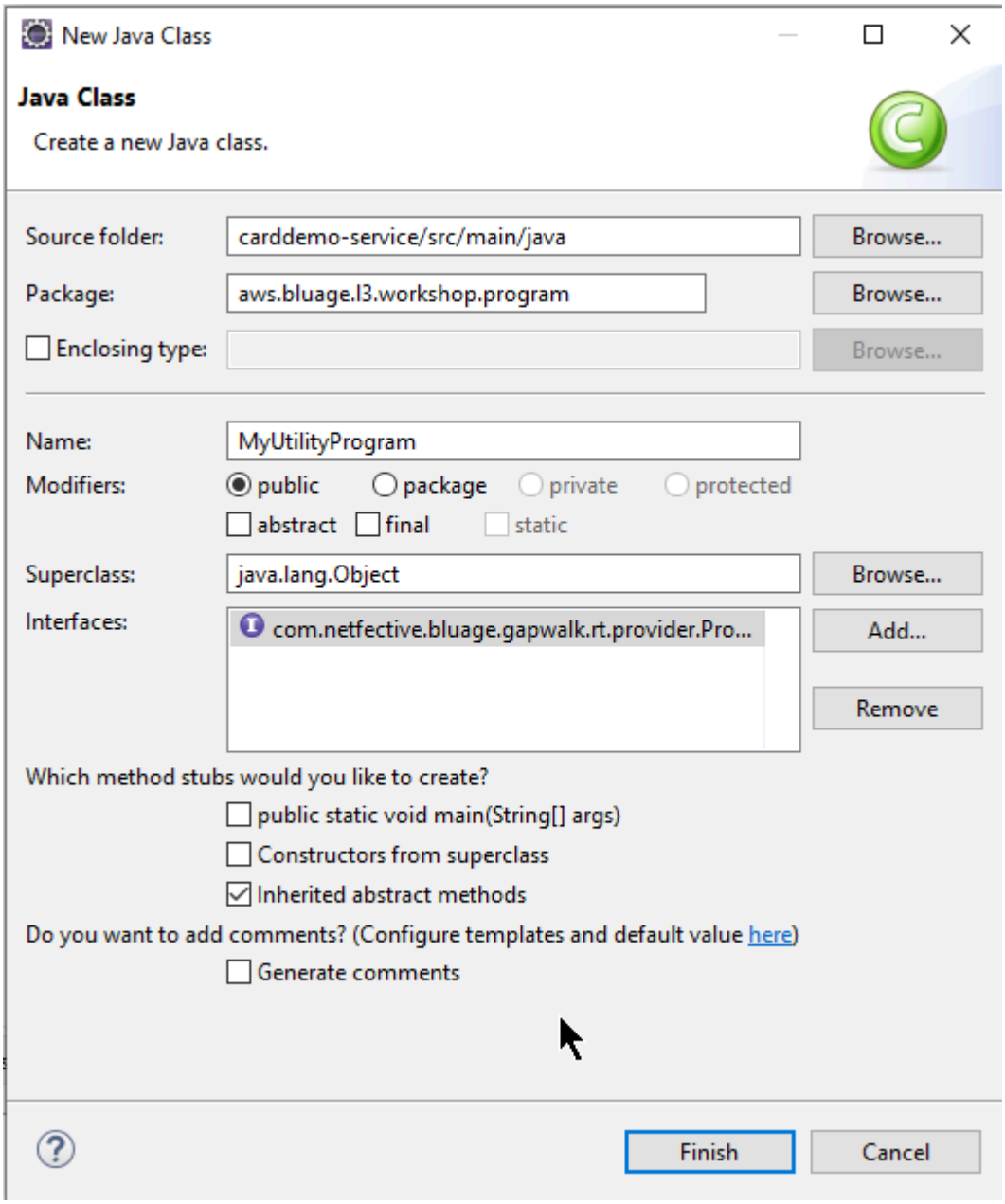
在這種情況下，您可能必須自己用 java 編寫丟失的程序（假設您對程序預期行為應該是什麼有足夠的了解，程序參數的內存佈局（如果有的話）。您的 java 程式必須符合本文件中描述的程式機制，以便其他程式和指令碼可以執行它。

若要確保程式可用，您必須完成兩個強制步驟：

- 編寫一個正確實現接Program口的類，以便它可以被註冊和調用。
- 確保您的程序已正確註冊，以便從其他程序/腳本中可見。

編寫程序實施

使用您IDE的創建一個實現Program接口的新 Java 類：



下圖顯示了 EclipseIDE，它需要創建要實現的所有強制性方法的護理：

MyUtilityProgram.java

```java
package aws.bluage.l3.workshop.program;

import java.util.Set;

public class MyUtilityProgram implements Program {

    @Override
    public ConfigurableApplicationContext getSpringApplication() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public Set<String> getProgramIdentifiers() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public Context getContext() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public void run(ExecutionController ctrl) {
        // TODO Auto-generated method stub

    }

}
```

彈簧整合

首先，類必須聲明為 Spring 組件。用@Component註釋註釋類：

```java
import org.springframework.context.ConfigurableApplicationContext;
import org.springframework.stereotype.Component;

import com.netfective.bluage.gapwalk.rt.call.ExecutionController;
import com.netfective.bluage.gapwalk.rt.context.Context;
import com.netfective.bluage.gapwalk.rt.provider.Program;

import aws.bluage.l3.workshop.SpringBootLauncher;

@Component
public class MyUtilityProgram implements Program {
```

接下來，正確實施所需的方法。在此範例的內容中，我們將新增MyUtilityProgram至已包含所有現代化程式的套件中。該放置允許程序使用現有的 Springboot 應用程序來提供所需ConfigurableApplicationContext的 getSpringApplication 方法實現：

```
public class MyUtilityProgram implements Program {

    @Override
    public ConfigurableApplicationContext getSpringApplication() {
        return SpringBootLauncher.getCac();
    }
```

您可以為自己的程序選擇不同的位置。例如，您可以在另一個專用服務項目中找到給定的程序。確保給定的服務項目有自己的 Springboot 應用程序，這使得它可以檢索 ApplicationContext （應該是一個`ConfigurableApplicationContext`）。

給一個身份的程序

要被其他程序和腳本調用，該程序必須給予至少一個標識符，這不能與系統中的任何其他現有的註冊程序發生衝突。識別碼的選擇可能是由於需要涵蓋現有舊版程式替換的需求所驅動；在這種情況下，您必須使用預期的識別碼，如同在整個舊版程式中發現的CALL事件中所遇到的那樣。在傳統系統中，大多數程序標識符都是 8 個字符。

在程序中創建一組不可修改的標識符是執行此操作的一種方法。下列範例顯示選擇 "MYUTILPG" 作為單一識別碼：

```
@Component
public class MyUtilityProgram implements Program {

    /**
     * Unique identifiers for the contained program.
     */
    private static final Set<String> programIdentifiers = Collections.unmodifiableSet(Stream.of("MYUTILPG").collect(Collectors.toSet()));


    public ConfigurableApplicationContext getSpringApplication() {

    @Override
    public Set<String> getProgramIdentifiers() {
        return programIdentifiers;
    }
```

將方案與前後關聯產生關聯

該程序需要一個伴隨`RuntimeContext`實例。對於現代化程序， AWS Blu Age 使用屬於舊計劃一部分的數據結構自動生成伴隨上下文。

如果您正在編寫自己的程序，則還必須編寫伴隨上下文。

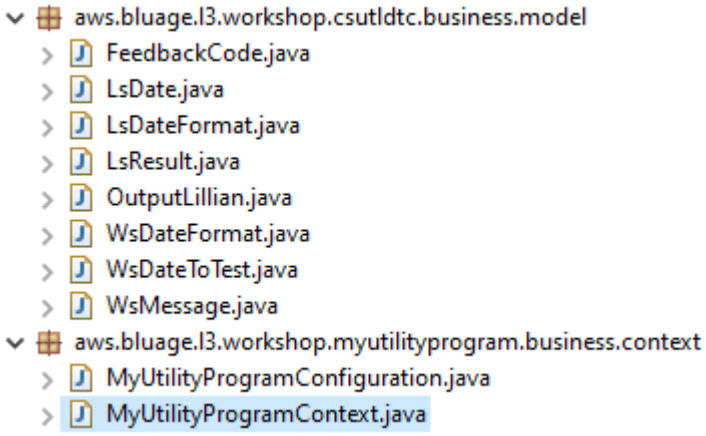
參考一下課程相關課程，您可以看到一個程序至少需要兩個同伴課程：

- 一個配置類。
- 使用配置的上下文類。

如果實用程序使用任何額外的數據結構，它應該被寫入，以及由上下文使用。

這些類應該在一個包中，該軟件包層次結構的一部分將在應用程序啟動時進行掃描，以確保上下文組件和配置將由 Spring 框架處理。

讓我們在實體項目中新創建的*base package*.myutilityprogram.business.context包中編寫一個最小的配置和上下文：



以下是配置內容。它使用的配置構建類似於附近的其他現代化程序。您可能必須根據您的特定需求進行自定義。

MyUtilityProgramConfiguration.java

```java
package aws.bluage.l3.workshop.myutilityprogram.business.context;

import java.nio.charset.Charset;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Lazy;

import com.netfective.bluage.gapwalk.datasimplifier.configuration.Configuration;
import com.netfective.bluage.gapwalk.datasimplifier.configuration.ConfigurationBuilder;

/**
 * Creates Datasimplifier configuration for the MyUtilityProgram context.
 */
@org.springframework.context.annotation.Configuration
@Lazy
public class MyUtilityProgramConfiguration {

    @Bean(name = "MyUtilityProgramContextConfiguration")
    public Configuration configuration() {
        return new ConfigurationBuilder()
                .encoding(Charset.forName("CP1047"))
                .humanReadableEncoding(Charset.forName("ISO-8859-15"))
                .initDefaultByte(0)
                .build();
    }
}

```

備註：

- 一般命名慣例為「ProgramName組態」。
- 它必須使用 @org. 彈簧框架. 上下文. 註釋. @Lazy
- bean 名稱通常遵循ProgramNameContextConfiguration 慣例，但這不是強制性的。確保避免整個項目中的 bean 名稱衝突。
- 要實現的單個方法必須返回一個`Configuration`對象。使用`ConfigurationBuilder`流利API來幫助您構建一個。

和相關的上下文：

MyUtilityProgramContext.java

```
2
3  import org.springframework.beans.factory.annotation.Qualifier;
4  import org.springframework.context.annotation.Lazy;
5  import org.springframework.context.annotation.Scope;
6  import org.springframework.stereotype.Component;
7
8  import com.netfective.bluage.gapwalk.datasimplifier.configuration.Configuration;
9  import com.netfective.bluage.gapwalk.rt.context.RuntimeContext;
10
11 @Component("aws.bluage.l3.workshop.mytuilityprogram.business.context.MyUtilityProgramContext")
12 @Lazy
13 @Scope("prototype")
14 public class MyUtilityProgramContext extends RuntimeContext{
15
16     protected MyUtilityProgramContext(@Qualifier("MyUtilityProgramContextConfiguration") Configuration configuration) {
17         super(configuration);
18     }
19
20     @Override
21     public void cleanUp() {
22         // TODO implement clean-up of associated data structures if any
23     }
24
25     @Override
26     protected void doReset() {
27         // TODO implement reset of associated data structures if any
28     }
29
30 }
31
```

備註

- 上下文類應該擴展現有的`Context`接口實現（`RuntimeContext`或者`JicsRuntimeContext`，這是一個JICS具`RuntimeContext`有特定項目的增強）。
- 一般命名慣例是ProgramName上下文。
- 您必須將其宣告為原型元件，並使用 @Lazy 註解。
- 構造函數引用相關的配置，使用 @Qualifier 註釋來定位適當的配置類。
- 如果實用程序使用一些額外的數據結構，它們應該是：

- 寫入並添加到*base package*.business.model包中。
- 在上下文中引用。看看其他現有的上下文類，看看如何引用數據串類，並根據需要調整上下文方法（構造函數/清理/重置）。

現在有專用上下文可用，讓新程序使用它：

MyUtilityProgram.java

```
19 import aws.bluage.l3.workshop.SpringBootLauncher;
20
21 @Component
22 @Import({
23     aws.bluage.l3.workshop.myutilityprogram.business.context.MyUtilityProgramConfiguration.class,
24     aws.bluage.l3.workshop.myutilityprogram.business.context.MyUtilityProgramContext.class
25 })
26 public class MyUtilityProgram implements Program {
27
28     @Autowired
29     BeanFactory beanFactory;
30
31     /**
32      * Unique identifiers for the contained program.
33      */
34     private static final Set<String> programIdentifiers = Collections.unmodifiableSet(Stream.of("MYUTILPG").collect(Collectors.toSet()));
35
36     private static final String programIdentifier = "MYUTILPG";
37
38     @Override
39     public ConfigurableApplicationContext getSpringApplication() {
40         return SpringBootLauncher.getCac();
41     }
42
43     @Override
44     public Set<String> getProgramIdentifiers() {
45         return programIdentifiers;
46     }
47
48     /**
49      * {@inheritDoc}
50      */
51     @Override
52     public String getProgramMainIdentifier() {
53         return programIdentifier;
54     }
55
56
57     @Override
58     public Context getContext() {
59         return ProgramContextStore.getOrCreate(
60         getProgramMainIdentifier(),
61         aws.bluage.l3.workshop.myutilityprogram.business.context.MyUtilityProgramContext.class,
62         beanFactory);
63     }
64
```

備註：

- 該 getContext 方法必須嚴格執行，如圖所示，使用對`ProgramContextStore`類的 getOrCreate 方法和 auto 有線 Spring 的委託`BeanFactory`。單個程序標識符用於存儲在程序上下文`ProgramContextStore`; 這個標識符被引用為「程序主標識符」。
- 伴隨配置和上下文類必須使用 `@Import` spring 註釋引用。

實作業務邏輯

當程式架構完成時，請實行新公用程式的企業邏輯。

在程序的run方法中執行此操作。此方法將在任何時候程序被調用，無論是由另一個程序或由腳本執行。

快樂的編碼！

處理程序註冊

最後，請確定新程式已在中正確註冊ProgramRegistry。如果您將新程序添加到已包含其他程序的軟件包中，則無需執行任何操作。在應用程序啟動時，將新程序拾取並註冊其所有鄰居程序。

如果您為程序選擇了其他位置，則必須確保該程序在 Tomcat 啟動時正確註冊。有關如何執行此操作的一些靈感，請查看服務項目中生成的 SpringbootLauncher 類的初始化方法（請參閱服務項目內容）。

檢查 Tomcat 的啟動日誌。每個程序註冊都會記錄下來。如果您的程序已成功註冊，您將找到匹配的日誌條目。

當您確定您的程序已正確註冊時，可以開始迭代業務邏輯編碼。

完整名稱對映

本節包含 AWS Blu Age 和第三方完整名稱對應的清單，可用於現代化的應用程式。

AWS 藍色時代完整名稱對應

短名稱	完整名稱
CallBuilder	com.netfective.bluage.gapwalk.runtime.statements.CallBuilder
Configuration	com.netfective.bluage.gapwalk.datasimplifier.configuration.Configuration
ConfigurationBuilder	com.netfective.bluage.gapwalk.datasimplifier.configuration.ConfigurationBuilder

短名稱	完整名稱
ExecutionController	com.netfective.bluage.gapwalk.rt.call.ExecutionController
ExecutionControllerImpl	com.netfective.bluage.gapwalk.rt.call.internal.ExecutionControllerImpl
File	com.netfective.bluage.gapwalk.rt.io.File
MainProgramRunner	com.netfective.bluage.gapwalk.rt.call.MainProgramRunner
Program	com.netfective.bluage.gapwalk.rt.provider.Program
ProgramContextStore	com.netfective.bluage.gapwalk.rt.context.ProgramContextStore
ProgramRegistry	com.netfective.bluage.gapwalk.rt.provider.ProgramRegistry
Record	com.netfective.bluage.gapwalk.datasimplifier.data.Record
RecordEntity	com.netfective.bluage.gapwalk.datasimplifier.entity.RecordEntity
RuntimeContext	com.netfective.bluage.gapwalk.rt.context.RuntimeContext
SimpleStateMachineController	com.netfective.bluage.gapwalk.rt.statemachine.SimpleStateMachineController

短名稱	完整名稱
StateMachineController	com.netfective.bluage.gapwalk.rt.statemachine.StateMachineController
StateMachineRunner	com.netfective.bluage.gapwalk.rt.statemachine.StateMachineRunner

第三方完整名稱對應

短名稱	完整名稱
@Autowired	org.springframework.beans.factory.annotation.Autowired
@Bean	org.springframework.context.annotation.Bean
BeanFactory	org.springframework.beans.factory.BeanFactory
@Component	org.springframework.stereotype.Component
ConfigurableApplicationContext	org.springframework.context.ConfigurableApplicationContext
@Import	org.springframework.context.annotation.Import
@Lazy	org.springframework.context.annotation.Lazy

AWS 藍光時代的數據簡化器是什麼

在大型主機和中階系統上 (以下主題稱為「傳統」系統) 上，常用的程式設計語言，例如 COBOL PL/I 或RPG提供記憶體的低階存取。此存取權限著重於透過區域、封裝或字母數字等原生類型存取的記憶體配置，也可能透過群組或陣列彙總。

通過類型字段和直接訪問字節（原始內存）對給定內存的混合訪問，並在給定程序中共存。例如，COBOL程序將以連續的字節集（ ）或以相同的方式（記錄LINKAGE）將參數傳遞給呼叫者，或者以相同的方式（記錄）將文件讀/寫數據傳遞給呼叫者，同時使用在抄本中組織的類型字段來解釋此類內存範圍。

這種對內存的原始和結構化訪問，依賴精確的字節級內存佈局以及諸如劃或打包等舊版類型的組合都是 Java 編程環境中既不是本地也不容易使用的功能。

作為將舊版程序現代化為 Java 的 AWS Blu Age 解決方案的一部分，Data S implifier 庫為現代化的 Java 程序提供了這種構造，並以 Java 開發人員盡可能熟悉的方式（獲取器/設置者，字節數組，基於類）公開這些構造。它是從這樣的程序生成的現代化 Java 代碼的核心依賴關係。

為了簡單起見，以下大多數說明都基於COBOL構造，但是您可以對PL1和RPG數據佈局現代化使API用相同的內容，因為大多數概念都是相似的。

主題

- 主要類別
- 數據綁定和訪問
- FQN所討論的 Java 類型

主要類別

為了方便閱讀，本文檔使用了 AWS Blu Age API 接口和類的 Java 簡稱。如需詳細資訊，請參閱FQN所討論的 Java 類型。

低階記憶體表示

在最低級別，內存（以快速，隨機的方式訪問的連續字節範圍）由`Record`接口表示。這個接口本質上是固定大小的字節數組的抽象。因此，它提供了能夠訪問或修改基礎字節的 setter 和 getter。

結構化數據表示

為了表示結構化數據，例如「01 數據項」或「01 字帖」，如中所找到的 COBOL DATADIVISION，使用類的子`RecordEntity`類。這些通常不是手工編寫的，而是由相應的遺留結構中的 AWS Blu Age 現

代化工具生成的。了解它們的主要結構仍然很有用API，因此您可以了解現代化程序中的代碼如何使用它們。在的情況下COBOL，該代碼是從它們生成的 Java PROCEDURE DIVISION。

產生的程式碼代表每個具有RecordEntity子類別的「01 資料項目」；每個基本欄位或彙總會表示為私有 Java 欄位，組織為樹狀結構 (除了根項目之外，每個項目都有父項目)。

為了說明目的，這裡是一個示例COBOL數據項，其次是相應的 AWS Blu Age 生成的代碼，該代碼將其現代化：

```
01 TST2.
 02 FILLER PIC X(4).
 02 F1     PIC 9(2) VALUE 42.
 02 FILLER PIC X.
 02        PIC 9(3) VALUE 123.
 02 F2     PIC X VALUE 'A'.
```

```
public class Tst2 extends RecordEntity {

    private final Group root = new Group(getData()).named("TST2");
    private final Filler filler = new Filler(root,new AlphanumericType(4));
    private final Elementary f1 = new Elementary(root,new ZonedType(2, 0, false),new
 BigDecimal("42")).named("F1");
    private final Filler filler1 = new Filler(root,new AlphanumericType(1));
    private final Filler filler2 = new Filler(root,new ZonedType(3, 0, false),new
 BigDecimal("123"));
    private final Elementary f2 = new Elementary(root,new
 AlphanumericType(1),"A").named("F2");



    /**
     * Instantiate a new Tst2 with a default record.
     * @param configuration the configuration
     */
    public Tst2(Configuration configuration) {
        super(configuration);
        setupRoot(root);
    }
    /**
     * Instantiate a new Tst2 bound to the provided record.
     * @param configuration the configuration
     * @param record the existing record to bind
     */
    public Tst2(Configuration configuration, RecordAdaptable record) {
```

```
        super(configuration);
        setupRoot(root, record);
    }

    /**
     * Gets the reference for attribute f1.
     * @return the f1 attribute reference
     */
    public ElementaryRangeReference getF1Reference() {
        return f1.getReference();
    }

    /* *
     * Getter for f1 attribute.
     * @return f1 attribute
     */
    public int getF1() {
        return f1.getValue();
    }

    /**
     * Setter for f1 attribute.
     * @param f1 the new value of f1
     */
    public void setF1(int f1) {
        this.f1.setValue(f1);
    }
    /**
     * Gets the reference for attribute f2.
     * @return the f2 attribute reference
     */
    public ElementaryRangeReference getF2Reference() {
        return f2.getReference();
    }

    /**
     * Getter for f2 attribute.
     * @return f2 attribute
     */
    public String getF2() {
        return f2.getValue();
    }

    /**
```

```
     * Setter for f2 attribute.
     * @param f2 the new value of f2
     */
    public void setF2(String f2) {
        this.f2.setValue(f2);
    }
}
```

基本領域

類別欄位 Elementary (或Filler未命名時) 代表舊資料結構的「葉」。它們與基礎字節的連續跨度（「範圍」）相關聯，並且通常具有一個類型（可能是參數化的），表示如何解釋和修改這些字節（通過分別「解碼」和「編碼」字節數組的值）。

所有基本類型都是的RangeType子類別。常見的類型有：

COBOL類型	資料簡化器類型
PIC X(n)	AlphanumericType
PIC 9(n)	ZonedType
PIC 9(n) COMP-3	PackedType
PIC 9(n) COMP-5	BinaryType

彙總欄位

聚合字段組織其內容的內存佈局（其他聚合或基本字段）。他們本身沒有基本類型。

Group字段表示內存中的連續字段。它們包含的每個字段在內存中以相同的順序佈置，第一個字段0相對於內存中的組字段位置處於偏移量，第二個字段處於偏移量0 + (size in bytes of first field)等。它們被用來表示相同包含COBOL字段下的字段序列。

Union字段表示訪問相同內存的多個字段。它們的每個包含的字段都在相對於存儲器中0的聯合字段位置的偏移位置進行佈局。例如，它們用來表示 COBOL "REDEFINES" 建構 (第一個 Union 子系是重新定義的資料項目，第二個子項是它的第一個重新定義等)。

數組字段（的子類Repetition）表示內存中其子字段的佈局的重複（無論是聚合本身還是基本項目）。他們在內存中佈置了給定數量的這樣的子佈局，每個都在偏移index * (size in bytes of child)。它們是用來表示 COBOL "OCCURS" 建構。

基元

在某些現代化的情況下，「原語」也可以用來呈現獨立的「根」數據項。這些在使用中非常相似，RecordEntity但不是來自它，也不是基於生成的代碼。相反，它們由 AWS Blu Age 運行時直接提供為接Primitive口的子類。此類提供的類別的範例為Alphanumeric或ZonedDecimal。

數據綁定和訪問

結構化數據和基礎數據之間的關聯可以通過多種方式完成。

用於此目的的一個重要接口是RecordAdaptable，用於獲取在RecordAdaptable基礎數據上Record提供「可寫視圖」。正如我們將在下面看到，多個類實現RecordAdaptable。相反地，AWS Blu Age APIs 和代碼操縱低級內存（例如程序參數，文件 I/O 記錄，CICS通信區域，分配的內存...）通常會期望 a RecordAdaptable 作為該內存的句柄。

在COBOL現代化的情況下，大多數數據項都與內存相關聯，這些內存將在相應的程序執行的生命週期內固定。為此，RecordEntity子類在生成的父對象（程序 Context）中實例化一次，並且將根據字節大小實例化它們的RecordEntity基礎Record。

在其他COBOL情況下，例如將LINKAGE元素與程序參數相關聯，或者將 F 構造現代化，RecordEntity實例必須與提供SETADDRESS的相關聯。RecordAdaptable為此，存在兩種機制：

- 如果RecordEntity實例已經存在，則可以使用該RecordEntity.bind(RecordAdaptable)方法（繼承自Bindable）來使該實例「指向」此RecordAdaptable。然後，在上調用的任何 getter 或 setter 都RecordEntity將由底層RecordAdaptable字節支持（字節讀取或寫入）。
- 如果要RecordEntity實例化，則可以使用接受 a RecordAdaptable 的生成構造函數。

相反，可以訪問Record當前綁定到結構化數據的。為此，RecordEntity實現RecordAdaptable，所getRecord()以可以在任何這樣的實例上調用。

最後，許多COBOL或CICS動詞需要訪問單個字段，用於閱讀或寫入目的。該RangeReference類用於表示這樣的訪問。它的實例可以從RecordEntity生成的getXXXReference()方法（XXX被訪問的字段）中獲取，並傳遞給運行時方法。 RangeReference通常用於訪問整個RecordEntity或Group，而其子類ElementaryRangeReference代表對Elementary字段的訪問。

請注意，上面的大多數觀察都適用於Primitive子類，因為它們努力實現與 AWS Blu Age 運行RecordEntity時提供的類似行為（而不是生成的代碼）。為了這個目的，Primitive實現的

所有子類ElementaryRangeReference和Bindable接口RecordAdaptable，以便可用於代替RecordEntity子類和基本字段。

FQN所討論的 Java 類型

下表顯示本節所討論之 Java 類型的完整名稱。

短名稱	完整名稱
Alphanumeric	com.netfective.bluage.gapwalk.datasimplifier.elementary.Alphanumeric
AlphanumericType	com.netfective.bluage.gapwalk.datasimplifier.metadata.type.AlphanumericType
BinaryType	com.netfective.bluage.gapwalk.datasimplifier.metadata.type.BinaryType
Bindable	com.netfective.bluage.gapwalk.datasimplifier.data.Bindable
Elementary	com.netfective.bluage.gapwalk.datasimplifier.data.structure.Elementary
ElementaryRangeReference	com.netfective.bluage.gapwalk.datasimplifier.entity.ElementaryRangeReference
Filler	com.netfective.bluage.gapwalk.datasimplifier.data.structure.Filler
Group	com.netfective.bluage.gapwalk.datasimplifier.data.structure.Group

短名稱	完整名稱
PackedType	com.netfective.bluage.gapwalk.datasimplifier.metadata.type.PackedType
Primitive	com.netfective.bluage.gapwalk.datasimplifier.elementary.Primitive
RangeReference	com.netfective.bluage.gapwalk.datasimplifier.entity.RangeReference
RangeType	com.netfective.bluage.gapwalk.datasimplifier.metadata.type.RangeType
Record	com.netfective.bluage.gapwalk.datasimplifier.data.Record
RecordAdaptable	com.netfective.bluage.gapwalk.datasimplifier.data.RecordAdaptable
RecordEntity	com.netfective.bluage.gapwalk.datasimplifier.entity.RecordEntity
Repetition	com.netfective.bluage.gapwalk.datasimplifier.data.structure.Repetition
Union	com.netfective.bluage.gapwalk.datasimplifier.data.structure.Union

短名稱	完整名稱
ZonedDecimal	com.netfective.bluage.gapwalk.datasimplifier.elementary.ZonedDecimal
ZonedType	com.netfective.bluage.gapwalk.datasimplifier.metadata.type.ZonedType

AWS 藍色時代布魯桑

在大型主機系統上 (下列主題稱為「舊版」)，業務資料通常會使用 VSAM (虛擬儲存存取方法) 來儲存。數據存儲在「記錄」（字節數組）中，屬於「數據集」。

有四個資料集組織：

- KSDS: 索引鍵排序資料集-記錄由主索引鍵 (不允許重複索引鍵) 以及選擇性的其他「備用」索引鍵。所有鍵值都是記錄字節數組的子集，每個鍵被定義為：
  - 偏移量（基於 0，0 是記錄字節數組內容的開始，以字節為單位）
  - 長度 (以位元組表示)
  - 它是否容忍重複的值
- ESDS: 入口排序數據集-記錄大多按順序訪問（根據其在數據集中的插入順序），但可以使用其他替代鍵進行訪問;
- RRDS：相對記錄數據集-記錄使用「跳躍」訪問，使用相對記錄號; 跳轉可以向前或向後進行;
- LDS：線性數據集-沒有記錄，只是一個字節流，組織在頁面中。主要用於舊版平台的內部用途。

使用 AWS Blu Age 重構方法將舊版應用程式現代化時，現代化的應用程式不再用於VSAM存取儲存的資料，同時保留資料存取邏輯。Blusam 組件就是答案：它允許從傳統數據集導出中導入數VSAM據，API為現代化的應用程序提供了一個與專用的管理 Web 應用程序一起操作它們。請參閱 the section called “藍色管理主控台”。

> **Note**
>
> 布魯薩姆只支援KSDSESDS、和。RRDS

Blusam API 可以保留資料存取邏輯 (循序、隨機和相對讀取；插入、更新和刪除記錄)，而依賴於混合快取策略和RDBMS基礎儲存的元件架構允許在有限資源的情況下進行高輸送量 I/O 作業。

布魯薩姆基礎設施

Blusam 依賴RDBMS於數據集存儲，無論是原始記錄數據和密鑰索引（如果適用）。最喜歡的選擇是使用 Postgre SQL 引擎，特別是使用 Amazon Aurora。本主題中的範例和插圖以此引擎為基礎。

其他支援的RDBMS引擎包括：

- Oracle
- Microsoft SQL 伺服器

注意

- 對於 AWS 大型主機現代化管理的環境，只有 Aurora Postgre SQL 引擎符合資格。在 Amazon EC2 部署上使用 AWS 藍光時代運行時，可以選擇其他RDBMS引擎。
- 在伺服器啟動時，Blusam 執行階段會檢查是否存在某些強制性技術資料表，如果找不到它們，則建立它們。因此，在組態中用來存取 Blusam 資料庫的角色必須被授與建立、更新和刪除資料庫表格的權限 (包括資料列和表格本身定義)。若要取得有關如何停用 Blusam 的資訊，請參閱。the section called “布魯薩姆配置”

快取

除了存儲本身，當耦合到緩存實現 Blusam 運行速度更快。

目前支援兩個快取引擎， EhCache 而 Redis 則各有自己的使用案例：

- EhCache ：獨立嵌入式揮發性本機快取
  - NOT符合 AWS 大型主機現代化管理環境部署的資格。
  - 通常在單個節點（例如單一 Apache Tomcat 服務器）用於運行現代化的應用程序時使用。例如，節點可能專用於主控批次工作任務。
  - 揮發性： EhCache 緩存實例是易失性的; 其內容將在服務器關閉時丟失。
  - 嵌入式： EhCache 和服務器共享相同的內JVM存空間（在定義託管計算機的規格時要考慮）。
- Redis 的：共享持久性緩存
  - 符合 AWS 大型主機現代化管理環境部署的資格。

- 通常用於多節點的情況下，特別是當多個伺服器位於負載平衡器後方時。快取內容會在所有節點之間共用。
- Redis 是持久的，並且不綁定到節點的生命週期。它在自己的專用機器或服務（例如，Amazon ElastiCache）上運行。快取對所有節點都是遠端的。

鎖定

為了處理數據集和記錄的並發訪問，Blusam 依賴於可配置的鎖定系統。鎖定可以應用於這兩個級別：數據集和記錄：

- 針對寫入目的鎖定資料集，可防止所有其他用戶端在任何層級 (資料集或記錄) 執行寫入作業。
- 在寫入記錄層級鎖定將防止其他用戶端僅對指定記錄執行寫入作業。

配置 Blusam 鎖定系統應根據緩存配置進行相應的操作：

- 如果 EhCache 選擇作為緩存實現，則不需要進一步的鎖定配置，因為應使用默認的內存鎖定系統。
- 如果選擇 Redis 作為緩存實現，則需要基於 Redis 的鎖定配置，以允許從多個節點並發訪問。用於鎖定的 Redis 快取不一定與用於資料集的快取相同。若要取得有關規劃以 Redis 為基礎的鎖定系統的資訊，請參閱。the section called “布魯薩姆配置”

布魯桑內在函數和從傳統數據遷移

存儲數據集：記錄和索引

匯入 Blusam 時，每個舊式資料集都會儲存到專用表格中；表格的每一列代表一筆記錄，使用兩欄：

- 數字 ID 列，大整數類型，也就是表的主鍵，用於存儲記錄的相對字節地址（RBA）。RBA代表從資料集的開頭以位元組為單位的偏移量，並從 0 開始。
- 位元組陣列記錄資料行，用來儲存原始記錄的內容。

例如，請參閱 CardDemo 應用程序中使用的KSDS數據集的內容：

```
SELECT * FROM public.aws_m2_carddemo_acctdata_vsam_ksds
ORDER BY id ASC
```

Data output | Messages | Notifications

	id [PK] bigint	record bytea
1	0	[binary data]
2	300	[binary data]
3	600	[binary data]
4	900	[binary data]
5	1200	[binary data]
6	1500	[binary data]
7	1800	[binary data]
8	2100	[binary data]
9	2400	[binary data]
10	2700	[binary data]
11	3000	[binary data]
12	3300	[binary data]
13	3600	[binary data]

- 這個特定的數據集具有固定的長度記錄，長度為 300 字節（因此 ID 的集合是 300 的倍數）。
- 預設情況下，用來查詢 postgre 資料SQL庫的 pgAdmin 工具不會顯示位元組陣列欄內容，而是列印 [二進位資料] 標籤。
- 原始記錄內容符合從舊版匯出的原始資料集，無需進行任何轉換。特別是，不會發生任何字元集轉換；這表示記錄的英數字元部分必須由使用舊字元集的現代化應用程式解碼，很可能是變體 EBCDIC。

關於數據集元數據和鍵索引：每個數據集都與名為表中的兩行相關聯`metadata`。這是預設的命名慣例。若要瞭解如何對其進行自訂，請參閱the section called “布魯薩姆配置”。

	name [PK] text	metadata oid
1	AWS_M2_CARDDEMO_ACCTDATA_VSAM_KSDS	66320
2	AWS_M2_CARDDEMO_ACCTDATA_VSAM_KSDS___internal	66321

- 第一列的資料集名稱做為 name 欄的值。metadata 資料行是二進位資料行，其中包含指定資料集之一般中繼資料的二進位序列化。如需詳細資訊，請參閱 the section called “一般資料集中繼資料屬性”。
- 第二列的資料集名稱以尾碼`__internal'`做為 name 欄的值。中繼資料行二進位內容取決於資料集的「權重」。
  - 對於中小型資料集，內容是以下項目的壓縮序列化：
    - 定義資料集所使用的索引鍵；主索引鍵定義 (用於KSDS) 和替代索引鍵定義 (如果適用) (針對KSDS/ESDS)
    - 鍵索引（如果適用）（KSDS/ESDS具有備用鍵定義）：用於索引瀏覽記錄; 鍵索引將鍵值映射到記錄RBA的;
    - 記錄長度圖：用於記錄的順序/相對瀏覽;
  - 對於大型/非常大的資料集，內容是以下項目的壓縮序列化：
    - 定義資料集所使用的索引鍵；主索引鍵定義 (用於KSDS) 和替代索引鍵定義 (如果適用) (針對KSDS/ESDS)

此外，使用分頁機制儲存大型/超大型資料集索引 (如果適用)；索引頁二進位序列化會儲存為專用資料表的列 (每個資料集索引鍵一個資料表)。索引的每個頁面都存儲在一行中，具有以下列：

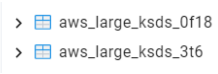
- id：索引頁面的技術標識符（數字主鍵）；
- firstkey：存儲在索引頁面中的第一個（最低）鍵值的二進制值；
- lastkey：存儲在索引頁面中最後一個（最高）鍵值的二進制值；
- metadata：索引頁面的二進位壓縮序列化 (將索引鍵值對應至記錄RBAs)。

	id [PK] bigint	firstkey bytea	lastkey bytea	metadata oid
1	1	[binary data]	[binary da...	6458928
2	2	[binary data]	[binary da...	6458929
3	3	[binary data]	[binary da...	6458930
4	4	[binary data]	[binary da...	6458931
5	5	[binary data]	[binary da...	6458932
6	6	[binary data]	[binary da...	6458933
7	7	[binary data]	[binary da...	6458934
8	8	[binary data]	[binary da...	6458935
9	9	[binary data]	[binary da...	6458936

資料表名稱是資料集名稱和金鑰內部名稱的串連，其中包含關於索引鍵的資訊，例如索引鍵偏移量、金鑰是否接受重複項目 (設定為 true 以允許重複項目)，以及金鑰長度。例如，假設名為 "AWS_ LARGE _KSDS" 的資料集具有下列兩個已定義的索引鍵：

- 主鍵 [偏移：0，重複：假，長度：18]
- 備用鍵 [偏移量：3，重複：真，長度：6]

在這種情況下，下表存儲與兩個鍵相關的索引。



使用寫入機制最佳化 I/O 輸送量

為了優化插入/更新/刪除操作性能，Blusam 引擎依賴於可配置的寫入機制。該機制建立在使用批量更新查詢處理持續性操作的專用線程池上，以最大限度地提高對 Blusam 存儲的 I/O 吞吐量。

Blusam 引擎會收集應用程式在記錄上完成的所有更新作業，並建立要傳送給專用執行緒處理的記錄批次。然後，這些批次將被保存到 Blusam 存儲中，使用批量更新查詢，避免使用原子持久性操作，從而確保了網絡帶寬的最佳使用。

該機制使用可配置的延遲（默認為一秒鐘）和可配置的批量大小（默認為 10000 條記錄）。只要符合下列兩個條件中的第一個條件，就會立即執行建置持續性查詢：

- 設定的延遲已經過去，批號不是空的
- 要處理的批號中的記錄數達到設定的限制

若要瞭解如何設定寫入式機制，請參閱。the section called “可選屬性”

選擇適當的存儲方案

如上一節所示，數據集的存儲方式取決於它們的「權重」。但是，對於數據集來說，什麼被認為是小型，中型或大型？ 何時選擇分頁存儲策略而不是常規的策略？

該問題的答案取決於以下內容。

- 裝載將使用這些資料集之現代化應用程式之每個伺服器上的可用記憶體數量。
- 快取基礎結構上的可用記憶體數量 (如果有的話)。

當使用非分頁索引存儲方案時，完整的索引鍵索引和記錄大小集合將在數據集開放時間，為每個數據集加載到服務器內存中。此外，如果涉及快取，所有的資料集記錄可能會以一般方法預先載入快取中，這可能會導致快取基礎結構端的記憶體資源耗盡。

根據定義的鍵的數量，鍵值的長度，記錄數和同時打開的數據集的數量，可以粗略評估給定的已知用例消耗的內存量。

如需進一步了解，請參閱 the section called “估計給定數據集的內存佔用”。

布魯桑移民

為指定的資料集選取適當的儲存配置之後，必須透過移轉舊式資料集來填入 blusam 儲存。

為了實現這一目標，必須使用舊數據集的原始二進制導出，而不在導出過程中使用任何字符集轉換。從舊版系統傳輸資料集匯出時，請確定不要損壞二進位格式。例如，使用時強制執行二進位模式FTP。

原始二進制導出僅包含記錄。匯入機制不需要金鑰/索引匯出，因為匯入機制會即時重新計算所有金鑰/索引。

一旦資料集二進位匯出可用，將其移轉至 Blusam 的幾個選項可供選擇：

在 AWS 大型主機現代化管理環境中：

- 使用專用功能匯入資料集。請參閱 the section called “匯入應用程式的資料集”。

或

- 使用資料集大量匯入功能。請參閱 the section called “資料集定義參考” 和 the section called “資料集請求格式範例 VSAM”。

或

- 使用 groovy 腳本導入數據集，使用專用的加載服務。

注意

目前只能使用 groovy 指令碼ESDS在大型主機上匯入大型KSDS和大型現代化管理環境。

在 Amazon 上的 AWS 藍色時代運行時EC2：

- 使用 Blusam 管理主控台匯入資料集。

或

- 使用 groovy 腳本導入數據集，使用專用的加載服務。

使用 Groovy 腳本導入數據集

本節將幫助您編寫 groovy 腳本以將傳統數據集導入到 Blusam 中。

它從一些強制性進口開始：

```
import com.netfective.bluage.gapwalk.bluesam.BluesamManager
import com.netfective.bluage.gapwalk.bluesam.metadata.Key;
import com.netfective.bluage.gapwalk.rt.provider.ServiceRegistry
import java.util.ArrayList; //used for alternate keys if any
```

之後，對於要導入的每個數據集，代碼都是根據給定的模式構建的：

1. 建立或清除地圖物件
2. 用所需的屬性填充地圖（這會因數據集類型而異-請參閱下面的詳細信息）
3. 檢索服務註冊表中用於數據集類型的正確加載服務
4. 執行服務，使用 map 作為引數

您可以使用下列識別碼從服務登錄擷取 5 個服務實作：

- `"BluesamKSDSFileLoader"`: 適合中小尺寸 KSDS
- `"BluesamESDSFileLoader"`適合中小型/中型 ESDS
- `"BluesamRRDSFileLoader"`：對於 RRDS
- `"BluesamLargeKSDSFileLoader"`：適用於大 KSDS
- `"BluesamLargeESDSFileLoader"`：適用於大 ESDS

是否為KSDS/選擇常規與大型版本的服務，ESDS取決於數據集的大小以及您要應用的存儲策略。若要瞭解如何選擇適當的儲存策略，請參閱the section called “選擇適當的存儲方案”。

為了能夠成功地將數據集導入 Blusam，必須為加載服務提供適當的屬性。

常見屬性：

- 強制性 (適用於各種資料集)
  - 「bluesamManager"：預期值為 `applicationContext.getBean(BluesamManager.class)`
  - 「datasetName"：資料集的名稱，以字串形式
  - 「inFilePath": 舊式資料集匯出的路徑 (做為字串)
  - 「recordLength「：固定記錄長度或 0 表示可變記錄長度的數據集，作為整數
- 選用
  - 不支援大型資料集：
    - 「isAppend"：一個布爾標誌，表示導入正在追加模式下發生（將記錄附加到現有的 blusam 數據集）。
    - 「useCompression"：布林旗標，表示將使用壓縮來儲存中繼資料。
  - 僅適用於大型資料集：
    - 「indexingPageSizeInMb"：每個索引頁的大小 (以 MB 為單位)，資料集中每個索引鍵的大小 (以 MB 為單位)，做為嚴格正整數

資料集種類相依屬性：

- KSDS/大號KSDS:
  - mandatory
    - 「primaryKey"：主鍵定義，使用`com.netfective.bluage.gapwalk.bluesam.metadata.Key`構造函數調用。
  - 可選：
    - 「alternateKeys"：一個替代鍵定義的 List (`java.util.List`)，使用`com.netfective.bluage.gapwalk.bluesam.metadata.Key`構造函數調用構建。
- ESDS/大號ESDS:
  - 可選：
    - 「alternateKeys"：一個替代鍵定義的 List (`java.util.List`)，使用`com.netfective.bluage.gapwalk.bluesam.metadata.Key`構造函數調用構建。
- RRDS:
  - 沒有。

關鍵構造函數調用：

- new Key(int offset, int length): 創建一個 Key 對象，具有給定的鍵屬性（偏移量和長度），並且不允許重複。這個變體應該被用來定義一個主鍵。
- new Key(boolean allowDuplicates, int offset, int length): 創建一個 Key 對象，具有給定的鍵屬性（偏移量和長度）和允許標誌的重複項。

下面的 Groovy 示例說明了各種加載情況。

加載一個大的KSDS，有兩個備用鍵：

```
import com.netfective.bluage.gapwalk.bluesam.BluesamManager
import com.netfective.bluage.gapwalk.bluesam.metadata.Key;
import com.netfective.bluage.gapwalk.rt.provider.ServiceRegistry
import java.util.ArrayList;

// Loading a large KSDS into Blusam
def map = [:]
map.put("bluesamManager", applicationContext.getBean(BluesamManager.class));
map.put("datasetName", "largeKsdsSample");
map.put("inFilePath", "/work/samples/largeKsdsSampleExport");
map.put("recordLength", 49);
map.put("primaryKey", new Key(0, 18));
ArrayList altKeys = [new Key(true, 10, 8), new Key(false, 0, 9)]
map.put("alternateKeys", altKeys);
map.put("indexingPageSizeInMb", 25);
def service = ServiceRegistry.getService("BluesamLargeKSDSFileLoader");
service.runService(map);
```

加載可變記錄長度ESDS，沒有備用鍵：

```
import com.netfective.bluage.gapwalk.bluesam.BluesamManager
import com.netfective.bluage.gapwalk.bluesam.metadata.Key;
import com.netfective.bluage.gapwalk.rt.provider.ServiceRegistry

// Loading an ESDS into Blusam
def map = [:]
map.put("bluesamManager", applicationContext.getBean(BluesamManager.class));
map.put("datasetName", "esdsSample");
map.put("inFilePath", "/work/samples/esdsSampleExport");
map.put("recordLength", 0);
def service = ServiceRegistry.getService("BluesamESDSFileLoader");
service.runService(map);
```

可變記錄長度資料集匯出將包含必要的記錄解碼器 Word (RDW) 資訊，以允許在讀取時間分割記錄。

載入固定記錄長度RRDS：

```
import com.netfective.bluage.gapwalk.bluesam.BluesamManager
import com.netfective.bluage.gapwalk.bluesam.metadata.Key;
import com.netfective.bluage.gapwalk.rt.provider.ServiceRegistry

// Loading a RRDS into Blusam
def map = [:]
map.put("bluesamManager", applicationContext.getBean(BluesamManager.class));
map.put("datasetName", "rrdsSample");
map.put("inFilePath", "/work/samples/rrdsSampleExport");
map.put("recordLength", 180);
def service = ServiceRegistry.getService("BluesamRRDSFileLoader");
service.runService(map);
```

此外，還可以使用組態項目 (在應用程式 main.yml 組態檔中設定) 來微調匯入程序：

- `bluesam.fileLoading.commitInterval`：一個嚴格的正整數，定義常規ESDS/KSDS/RRDS導入機制的提交間隔。不適用於大型資料集匯入。預設值為 10 萬。

布魯薩姆配置

設定 Blusam 會發生在應用程式 main.yml 組態檔案 (或應用程式 Blusam 管理主控台----應用程式的獨立部署的應用程式 bac.yml 組態檔中)。BAC

Blusam 必須在兩個方面進行配置：

- Blusam 存儲和緩存訪問配置
- 布魯薩姆引擎配置

Blusam 存儲和緩存訪問配置

如需如何使用密碼管理員或資料來源設定 Blusam 儲存區和快取存取權的相關資訊，請參閱。the section called “AWS 藍色時代運行時配置”

注意

關於 Blusam 儲存體的存取，使用的認證將指向具有相關權限的連線角色。若要讓 Blusam 引擎能夠如預期般運作，連線角色必須具備下列權限：

- 連接到數據庫
- 創建/刪除/更改/截斷表和視圖
- 選擇/插入/刪除/更新表格和視圖中的行
- 執行函數或程序

布魯薩姆引擎配置

停用布魯薩姆支援

首先，讓我們提一下，通過將bluesam.disabled屬性設置為，可以完全禁用 Blusam 支持。true在應用程序啟動時，服務器日誌中將顯示信息消息，提醒 Blusam 禁用：

```
BLUESAM is disabled. No operations allowed.
```

在這種情況下，不需要有關 Blusam 的進一步配置，任何嘗試使用 Blusam 相關功能（以編程方式或通過REST調用）都會在 Java 代碼執行UnsupportedOperationException中引發一個關於 Blusam 的相關解釋消息被禁用。

布魯薩姆引擎屬性

Blusam 引擎組態屬性會重新分組在 Blueam key prefix 之下：

強制性質

- cache：使用所選緩存實現進行價值。有效的 值如下：
  - ehcache：適用於本機內嵌式 ehcache 使用。請參閱上述相關使用案例限制。
  - redis：用於共享遠程 Redis 緩存使用。這是 AWS 大型主機現代化管理使用案例的偏好選項。
  - none：若要停用儲存區快取
- persistence：與選擇的存儲引擎進行價值。有效的 值如下：
  - pgsql（對於波斯特雷SQL引擎：最低版本 10.0-推薦版本 > = 14.0）
  - oracle（對於甲骨文引擎：最小版本 19c）
  - mssql（對於SQL服務器引擎：所需的最低版本是SQL服務器 2019）
- 數據源引用：<persistence engine>.dataSource將指向與 Blusam 存儲的連接的定 dataSource 義，該存儲在配置文件中的其他位置定義。通常它被命名bluesamDs。

注意

每當 Redis 被用作緩存機制，無論是用於數據還是鎖定（見下文），對 Redis 實例的訪問都將被配置。如需詳細資訊，請參閱 the section called “可用的 Redis 的緩存屬性”。

可選屬性

布魯薩姆鎖定：屬性的前綴為 `locks`

- `cache`: 只有可用的值是`redis`，指定將使用基於 rediss 的鎖定機制（當 blusam 存儲緩存也是基於 rediss 時使用）。如果內容遺失或未設定為`redis`，則會改用預設的記憶體內鎖定機制。
- `lockTimeOut`：正長整數值，在嘗試鎖定已鎖定的元素標記為失敗之前，以毫秒為單位表示的超時值。預設為`500`。
- `locksDeadTime`: 正長整數值，代表應用程式可以保留鎖定的最大時間 (以毫秒為單位)。鎖定會在經過時間後自動標示為已過期並釋放。默認為`1000`;
- `locksCheck`：一個字符串，用於定義當前 blusam 鎖定管理器使用的鎖定檢查策略，關於過期的鎖定刪除。要在以下值中選擇：
  - `off`：不會執行任何檢查。氣餒，因為可能會發生死鎖。
  - `reboot`：檢查是在重新開機或應用程式啟動時執行。當時會釋放所有過期的鎖定。此為預設值。
  - `timeout`：會在重新開機或應用程式啟動時執行檢查，或在嘗試鎖定資料集期間逾時到期時執行。過期的鎖定會立即釋放。

寫入機制：屬性前綴為 key：`write-behind`

- `enabled`: `true` (預設值和建議值)`false`，或啟用或停用寫入機制。停用此機制會極大地影響寫入效能，因此不建議使用。
- `maxDelay`：要觸發的線程的最大持續時間。預設為 `"1s"` (一秒)。保留預設值通常是個好主意，除非特定條件需要調整此值。在任何情況下，該值應保持較低（在 3 秒以下）。延遲字符串的格式是：`<integer value><optional whitespace><time unit>`其中`<time unit>`是以下值中選擇：
  - `"ns"`: 納秒
  - `"µs"`: 微秒
  - `"ms"`：毫秒
  - `"s"`：秒

- threads：專用寫入執行緒的數目。預設為5。您需要根據運行 Blusam 引擎的主機的計算能力來調整此值。使用更高的值並不相關，希望性能提高，因為限制因素將成為處理眾多並發批次查詢的存儲RDBMS能力。建議值通常在 4-8 的範圍內。
- batchSize：一個正整數，代表了很多記錄的最大數量，這些記錄將被分派以進行批量處理到線程。其值必須介於 1 和 32767 之間。預設為10000。使用 1 as value 會破壞避免使用原子更新查詢的機制的目的; 要使用的合適的最小值就在附近。1000

嵌入式 EhCache 微調：屬性前綴ehcache為 key：

- resource-pool:
  - size：為嵌入式緩存分配的內存大小，以字符串表示。默認為"1024MB"（1 千兆字節）。根據託管 Blusam 引擎的機器的可用內存以及應用程序正在使用的數據集的大小進行調整。大小字符串的格式是：<integer value><optional whitespace><memory unit>其中<memory-unit>是以下值中選擇：
    - B：位元組
    - KB: 千字節
    - MB: 百萬位元組
    - GB: 千兆字節
    - TB: 兆位元組
  - heap：true或者false，指示緩存是否會消耗JVM堆內存。默認為true（緩存性能的最快選項，但緩存存儲佔用JVM堆內存中的RAM內存）。將此屬性設置為false將切換到非堆內存，這將是較慢的，由於與JVM堆所需的交換。

示例配置片段：

```
dataSource:
  bluesamDs:
    driver-class-name: org.postgresql.Driver
    ...
    ...
bluesam:
  locks:
    lockTimeOut: 700
  cache: ehcache
  persistence: pgsql
  ehcache:
    resource-pool:
```

```
      size: 8GB
  write-behind:
    enabled: true
    threads: 8
    batchsize: 5000
  pgsql:
    dataSource : bluesamDs
```

藍色管理主控台

Blusam 管理主控台 (BAC) 是一個 Web 應用程式，用來管理 Blusam 儲存裝置。如需有關的資訊 BAC，請參閱the section called “藍色管理主控台”。

附錄

一般資料集中繼資料屬性

通用數據集元數據序列化屬性列表：

- 名稱 (資料集的)
- 類型 (KSDS、大KSDSESDS、大ESDS或RRDS)
- 緩存預熱標誌（數據集是否應在服務器啟動時預加載緩存中）
- 壓縮用法旗標 (是否以壓縮格式或原始格式儲存記錄)
- 建立日期
- 上次修改日期
- 固定長度記錄旗標 (資料集記錄是否全部具有相同的長度)
- 記錄長度-僅對固定記錄長度有意義
- 頁面大小（用於在需要時自定義用於預加載緩存的分頁 sql 查詢）
- size（數據集的大小-記錄的累計長度）
- 最後一個偏移量（即添加到數據集RBA的最新記錄的偏移量）
- 下一個偏移量（下一個可用的偏移量，用於向數據集添加新記錄）
- 如果有意義，則定義資料集所使用的索引鍵；每個索引鍵均由其種類 (替代索引鍵集合的主要或一部分) 定義，以及三個屬性：
  - offset：位置在鍵值的起始字節的記錄;
  - length：以字節為單位的密鑰值的長度。因此，關鍵值是字節數組，它是從位置開始`key offset`和結束的記錄的子集`key offset + length - 1`;

- 允許重複標誌：密鑰是否接受重複項（設置為 true 以允許重複）。

估計給定數據集的內存佔用

對於中小型數據集，元數據（各種鍵的大小和索引）將完全加載到內存中。為裝載用於執行現代化應用程式之伺服器的機器配置適當的資源，需要計算出 Blusam 資料集引起的記憶體消耗量，尤其是中繼資料。本節為有關運營商提供實際答案。

給定的公式僅適用於 Blusam 中小型數據集，而不是使用「大」存儲策略。

布魯薩姆資料集中繼資料

對於 Blusam 資料集，中繼資料分為兩部分：

- 核心元數據：保存有關數據集的全局信息。與內部元數據相比，它的內存佔用可以被認為是可以忽略不計的。
- 內部中繼資料：保存記錄大小和索引鍵索引的相關資訊；當資料集不是空的時候，這就是載入裝載現代化應用程式的應用程式伺服器時，會耗用記憶體。以下各節詳細說明了消耗的內存如何隨著記錄數增長。

計算內部中繼資料足

記錄大小地圖

首先，內部元數據存儲一個映射來保存每條記錄的大小（作為整數）給定其RBA（相對字節地址-存儲為長數字）。

該數據結構的內存佔用量，以字節為單位：`80 * number of records`

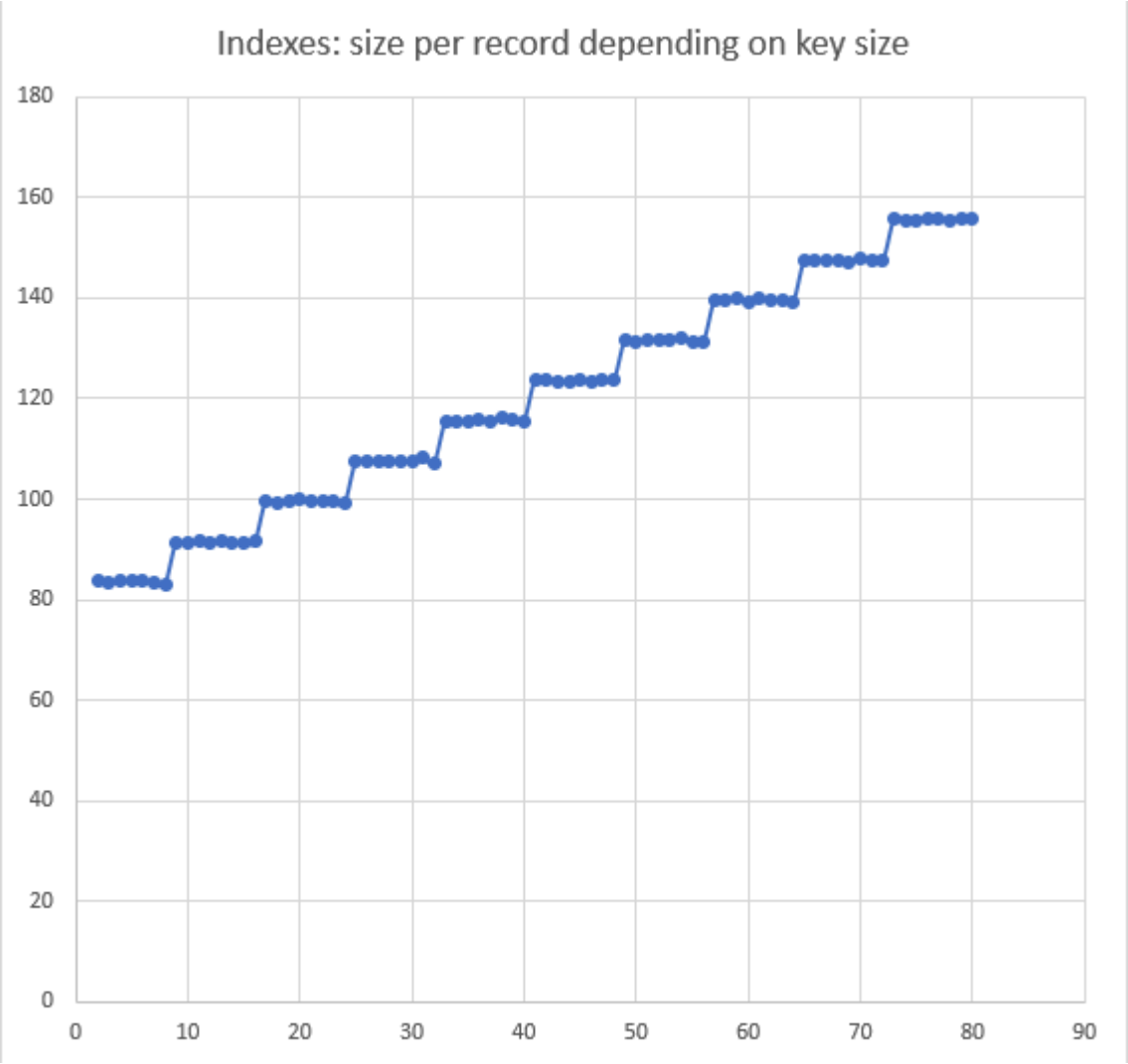
這適用於所有資料集種類。

索引

關於和上的主索引鍵KSDS或其他索引鍵的索引 ESDSKSDS，佔用空間的計算取決於兩個因素：

- 在數據集的記錄的數量;
- 密鑰的大小，以字節為單位。

下圖顯示根據金鑰大小 (x 軸) 的每個記錄 (y 軸) 的金鑰索引大小。



用於評估數據集的給定鍵索引的足跡的相應公式如下：

```
index footprint = number of records * ( 83 + 8 (key length / 8))
```

其中 '/' 代表整數除法。

範例：

- 資料集 1：
  - 記錄數目：
  - 金鑰長度 = 15，因此 (金鑰長度/8) = 1
  - 索引使用空間 = 459 996 * (83 + (8*1)) = 41 859 636 位元組 (相當於約 39 MB)
- 資料集 2：

- 記錄數目：
- 金鑰長度 = 18，因此 (金鑰長度/8) = 2
- 索引使用空間 = 13 095 783 * (83 + (8*2)) = 1 296 482 517 位元組 (大約相當於 1.2 GB)

指定資料集的總覆蓋區是所有索引鍵索引的所有外形圖的總和，以及記錄大小對映的覆蓋區。

例如，以僅具有單一索引鍵的範例資料集 2 為例，全域覆蓋區為：

- 記錄大小對應:
- 索引鍵索引：1 296 482 517 個位元組 (請參閱上文)
- 總佔用空間 = 2 344 145 157 個位元組 (等於約 2.18 GB)

AWS 藍色時代藍色管理主控台

Blusam 管理主控台 (BAC) 是處理 Blusam 資料集的安全網路應用程式。本指南涵蓋了使BAC用者介面。如需透過REST端點進行遠端管理，請參閱 Blusam 應用程式主控台REST端點。

主題

- 部署 BAC
- 使用 BAC
- LISTCATJSON格式

部署 BAC

可作為安全的單一 Web 應用程式使用，使用網頁封存格式 (.war)。BAC它旨在與 BluAge GapWalk 應用程序一起部署，在 Apache Tomcat 應用程序服務器中，但也可以部署為獨立應用程序。如果存在，則BAC繼承對 Blusam 存儲從 Gapwalk 應用程序配置的訪問。

BAC有它自己的專用組態檔案，名為`application-bac.yml`。如需組態詳細資訊，請參閱the section called “BAC專用配置文件”。

該BAC是安全的。如需安全性組態的詳細資訊，請參閱the section called “設定安全性 BAC”。

BAC專用配置文件

獨立部署：如果單獨部BAC署 GapWalk 應用程式，則必須在應用程式 bac.yml 組態檔中設定與 Blusam 儲存體的連線。

必須在組態檔案中設定用於瀏覽資料集記錄的資料集組態的預設值。請參閱 the section called “瀏覽資料集中的記錄”。記錄瀏覽頁面可以使用選擇性的遮罩機制，以便在記錄的內容上顯示結構化檢視。使用遮罩時，某些性質會影響記錄檢視。

必須在組態檔案中設定下列可配置屬性。BAC應用程式不會假設這些屬性的任何預設值。

金鑰	Type	描述
bac.crud.limit	integer	正整數值提供瀏覽記錄時傳回的最大記錄數目。使用0意味著無限。推薦值：10（然後根據瀏覽頁面上的數據設置調整數據設置的值，以滿足您的需求）。
bac.crud.encoding	string	默認字符集名稱，用於將記錄字節解碼為字母數字內容。提供的字符集名稱必須與 java 兼容（請參閱 Java 文檔以獲取支持的字符集）。建議值：在傳統平台上使用的舊字符集，其中數據集來自；這在大多數情況下都是EBCDIC變體。
bac.crud.initCharacter	string	用於初始化數據項的默認字符（字節）。可以使用兩個特殊值："LOW-VALUE" 0x00 位元組 (建議值) 和 "HI-VALUE" 0xFF 位元組。套用遮罩時使用。
bac.crud.defaultCharacter	string	默認字符（byte），作為一個字符串，用於填充記錄（右側）。建議值:" "(空格)。套用遮罩時使用。
bac.crud.blankCharacter	string	預設字元 (byte)，做為一個字元字串，用來表示記錄中的空

金鑰	Type	描述
		白。建議值:" "(空格)。套用遮罩時使用。
bac.crud.strictZoned	boolean	用於指示記錄使用哪個分區模式的旗標。如果true，將使用「嚴格」區域模式；如果false，將使用「已修改的區域」模式。建議值：true。套用遮罩時使用。
bac.crud.decimalSeparator	string	在編輯數字欄位中用作小數分隔符號的字元 (套用遮罩時使用)。
bac.crud.currencySign	string	套用格式化時，預設字元 (當套用遮罩時使用)，用來表示編輯數字欄位中的貨幣，做為一個字元字串。
bac.crud.pictureCurrencySign	string	默認字符，作為一個字符串，用於在數字編輯字段圖片中表示貨幣（應用掩碼時使用）。

下面的示例是一個配置文件片段。

```
bac.crud.limit: 10
bac.crud.encoding: ascii
bac.crud.initCharacter: "LOW-VALUE"
bac.crud.defaultCharacter: " "
bac.crud.blankCharacter: " "
bac.crud.strictZoned: true
bac.crud.decimalSeparator: "."
bac.crud.currencySign: "$"
bac.crud.pictureCurrencySign: "$"
```

設定安全性 BAC

配置的安全性BAC依賴於本文件頁面中詳述的機制。身份驗證方案是OAuth2，並提供 Amazon Cognito 或鍵盤遮罩的組態詳細資料。

雖然可以應用一般設置，但有關的一些細節BAC需要在這裡詳細說明。使用以角色為基礎的策略來保護BAC功能的存取權，並依賴下列角色。

- ROLE_USER:
  - 基本使用者角色
  - 不允許匯入、匯出、建立或刪除資料集
  - 無法控制快取政策
  - 不允許管理功能
- ROLE_ADMIN:
  - 繼承 ROLE _ USER 權限
  - 允許所有資料集作業
  - 允許快取原則管理

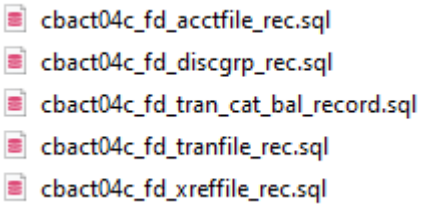
安裝口罩

在 Blusam 存儲中，數據集記錄存儲在數據庫中的字節數組列中，用於多功能性和性能考慮。具有訪問結構化視圖，使用字段，業務記錄，基於應用程序的觀點是一個方便的功能BAC. 這依賴於 BluAge 驅動的現代化過程中生產的SQL口罩。

對於要生成的SQL掩碼，請確保將 BluInsights 轉換中心配置中的相關選項（`export.SQL.masks`）設置為 true：

遮罩是可從特定專案下載的現代化成品 BluInsights的一部分。它們是由現代化程序組織的SQL腳本，給出了對數據集記錄的應用觀點。

例如，使用AWS CardDemo 示例應用程序，您可以在下載的工件從此應用程序的現代化結果中找到，以下SQL掩碼程序 CBACT 04C.CBL：



每個SQL遮罩名稱都是程式名稱的串連，以及程式中指定資料集的記錄結構名稱。

例如，查看 [CBACT04C.cbl 程序，給定的文件控制條目：

```
    FILE-CONTROL.
        SELECT TCATBAL-FILE ASSIGN TO TCATBALF
               ORGANIZATION IS INDEXED
               ACCESS MODE  IS SEQUENTIAL
               RECORD KEY   IS FD-TRAN-CAT-KEY
               FILE STATUS  IS TCATBALF-STATUS.
```

與給定的 FD 記錄定義相關聯

```
       FILE SECTION.
```

```
       FD  TCATBAL-FILE.
       01  FD-TRAN-CAT-BAL-RECORD.
           05 FD-TRAN-CAT-KEY.
              10 FD-TRANCAT-ACCT-ID             PIC 9(11).
              10 FD-TRANCAT-TYPE-CD             PIC X(02).
              10 FD-TRANCAT-CD                  PIC 9(04).
           05 FD-FD-TRAN-CAT-DATA               PIC X(33).
```

命名的匹配SQL掩碼cbact04c_fd_tran_cat_bal_record.SQL是給出了程序 CBACT 04C.CBL 上命名的 FD 記錄的點的掩碼。FD-TRAN-CAT-BAL-RECORD

它的內容是：

```
-- Generated by Blu Age Velocity
-- Mask : cbact04c_fd_tran_cat_bal_record

INSERT INTO mask (name, length) VALUES ('cbact04c_fd_tran_cat_bal_record', 50);
  INSERT INTO mask_item (name, c_offset, length, skip, type, options, mask_fk) VALUES
 ('fd_trancat_acct_id', 1, 11, false, 'zoned', 'integerSize=11!fractionalSize=0!
signed=false', (SELECT MAX(id) FROM mask));
  INSERT INTO mask_item (name, c_offset, length, skip, type, options, mask_fk) VALUES
 ('fd_trancat_type_cd', 12, 2, false, 'alphanumeric', 'length=2', (SELECT MAX(id) FROM
 mask));
  INSERT INTO mask_item (name, c_offset, length, skip, type, options, mask_fk)
 VALUES ('fd_trancat_cd', 14, 4, false, 'zoned', 'integerSize=4!fractionalSize=0!
signed=false', (SELECT MAX(id) FROM mask));
  INSERT INTO mask_item (name, c_offset, length, skip, type, options, mask_fk) VALUES
 ('fd_fd_tran_cat_data', 18, 33, false, 'alphanumeric', 'length=33', (SELECT MAX(id)
 FROM mask));
```

掩碼存儲在使用兩個表的 Blusam 存儲：

- 遮罩：用於識別口罩。MAS 表格的欄位如下：
  - name：用於存儲掩碼標識（用作主鍵，所以必須是唯一的）
  - length：記錄遮罩的大小 (以位元組為單位)
- 遮罩項目：用於存儲口罩的詳細信息。從 FD 記錄定義的每個基本字段將產生在 mask_item 表中的行，與如何解釋給定的記錄部分的詳細信息。遮罩項目表格的欄位如下：
  - name：記錄字段的名稱，基於基本名稱，使用小寫並用下劃線替換破折號
  - c_offset：1 基於記錄子部分的偏移量，用於字段內容
  - length：記錄子部分的字節長度，用於字段內容

- skip：標誌，指示是否應該跳過給定的記錄部分，在視圖演示文稿
- type：欄位種類 (根據其舊版圖片子句)
- 選項：其他類型選項-類型依賴
- mask_fk：參考要附加此項目的遮罩識別碼

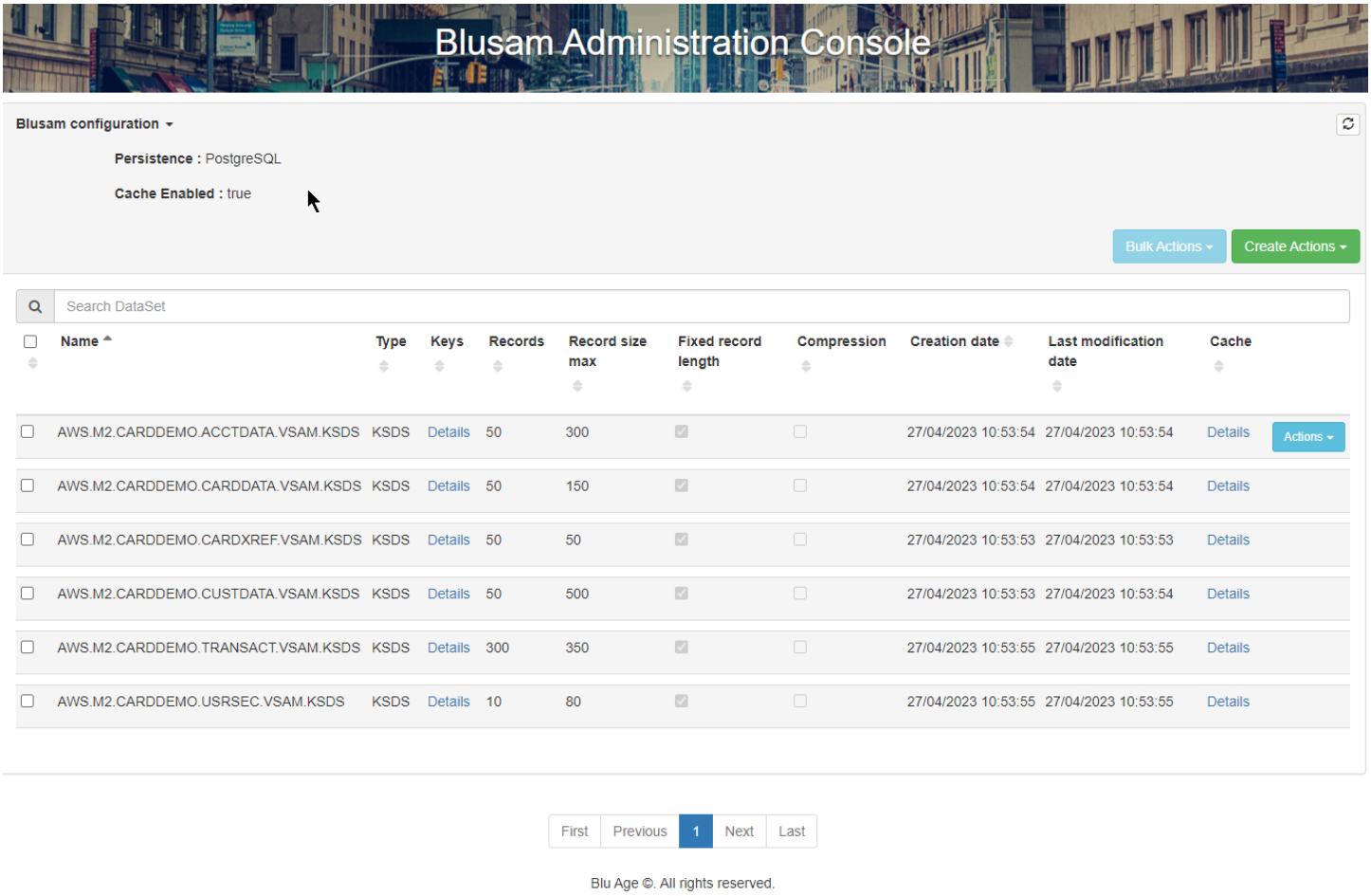
注意下列事項：

- SQLmask 代表程式對資料集中記錄的某個觀點：數個程式在指定的資料集上可能具有不同的觀點；請僅安裝您找到與您目的相關的遮罩。
- SQL遮罩也可以代表從程式的檢視點，根據WORKINGSTORAGE部分的 01 資料結構，不僅來自 FD 記錄。SQL遮罩會根據其性質組織成子資料夾：
  - 基於 FD 記錄的掩碼將位於命名的子文件夾 `file`
  - 01 基於數據結構的掩碼將位於名為的子文件夾中 `working`

  雖然 FD 記錄定義始終與資料集中的記錄內容相符，但 01 個資料結構可能不會對齊，或者只代表資料集記錄中的子集。在使用它們之前，請檢查代碼並理解可能的缺點。

使用 BAC

由BAC於受到保護並根據使用者角色提供使用功能的權限，因此存取應用程式的第一個步驟是驗證您自己。驗證步驟後，您將被重定向到主頁。首頁會顯示在 Blusam 儲存體中找到的資料集分頁清單：



若要返回列出資料集的首頁，請選擇應用程式任何頁面左上角的 BluAge 標誌。下圖顯示了徽標。

標示為「BluSam config」的可折疊式接頭包含有關使用 BluSam 儲存配置的資訊：

- `Persistence`：持久性存儲選擇（這裡 PostgreSQL）
- `Cache Enabled`：是否已啟用儲存快取

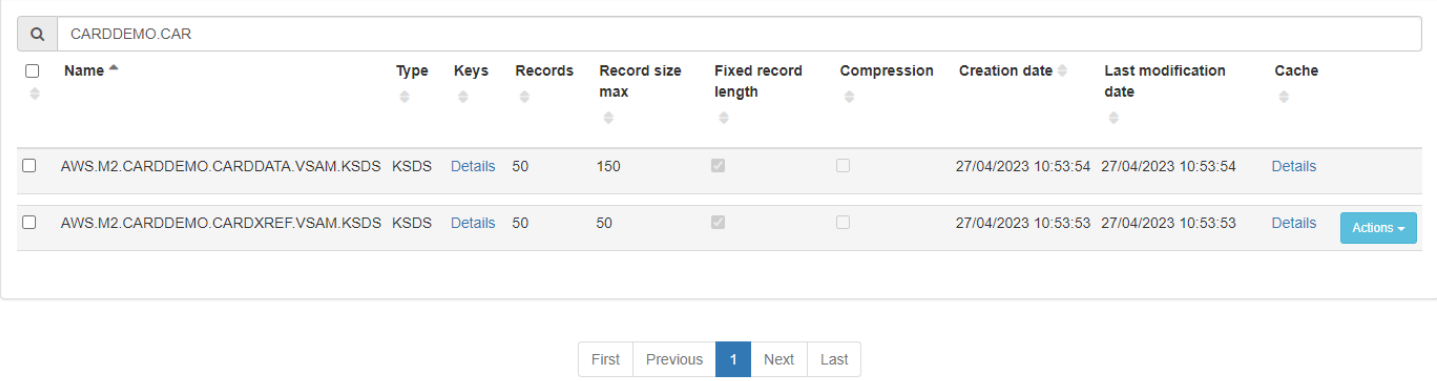
在標題的右側，有兩個下拉列表，每個列出與數據集相關的操作：

- 大量動作
- 建立動作

若要瞭解這些清單的詳細內容，請參閱the section called “現有的資料集作業”。

未選取任何資料集時，會停用 「批次處理動作」 按鈕。

您可以使用搜尋欄位，根據資料集名稱來篩選清單：



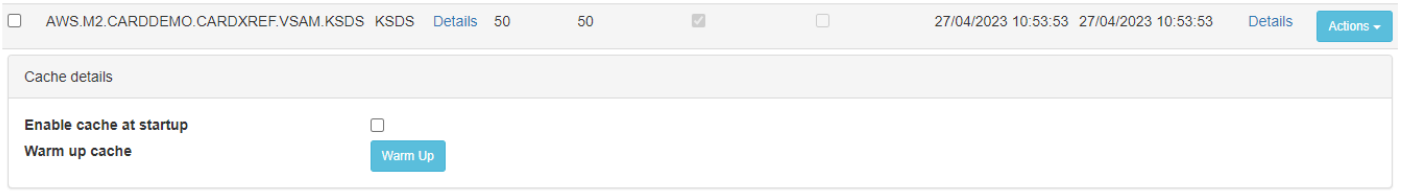
接下來的分頁清單會顯示每個表格資料列一個資料集，其中包含下列欄：

- 選取核取方塊：用於選取目前資料集的核取方塊。
- 名稱：資料集的名稱。
- 類型：資料集的類型，下列其中一項：
  - KSDS
  - ESDS
  - RRDS
- 金鑰：顯示或隱藏金鑰詳細資料的連結 (如果有的話)。例如，給定的KSDS具有強制性的主鍵和一個替代鍵。

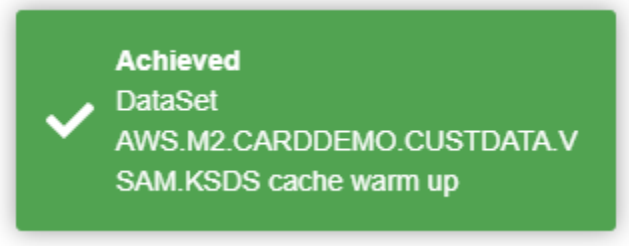
每個索引鍵有一個資料列，其中包含下列欄。沒有任何欄位是可編輯的。

- 鍵性質：主鍵或替代鍵
- 名稱：金鑰的名稱
- 唯一：密鑰是否接受重複的條目
- 偏移量：記錄內鍵開始的偏移量
- 長度：記錄中索引鍵部分的長度 (以位元組為單位)

- 記錄：資料集中的記錄總數。
- 記錄大小 max：記錄的最大大小，以字節表示。
- 固定記錄長度：指出記錄是固定長度 (已選取) 還是可變長度 (未選取) 的核取方塊。
- 壓縮：指出壓縮是否套用 (已選取) 或未 (未選取) 至儲存的索引的核取方塊。
- 建立日期：在 Blusam 儲存裝置中建立資料集的日期。
- 上次修改日期：上次在 Blusam 儲存裝置中更新資料集的日期。
- 快取：顯示或隱藏套用至此資料集之快取策略詳細資訊的連結。



  - 啟動時啟用快取：用於指定此資料集之啟動快取策略的核取方塊。如果選取此選項，資料集會在啟動時載入快取中。
  - 暖機快取：將指定資料集載入快取的按鈕，立即啟動 (但保留快取需要一些時間，具體取決於資料集大小和金鑰數目)。數據集被加載到緩存後，出現類似下面的通知。

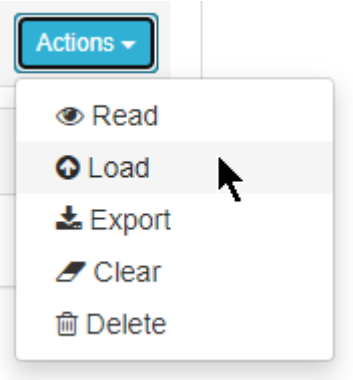


- 動作：可能的資料集作業的下拉式清單。如需詳細資訊，請參閱 the section called “現有的資料集作業”。

在頁面底部，有一個常規的分頁導航小部件，用於瀏覽數據集列表的頁面。

現有的資料集作業

針對分頁清單中的每個資料集，都會有一個「動作」(A ctions) 下拉式清單，其中包含下列內容：



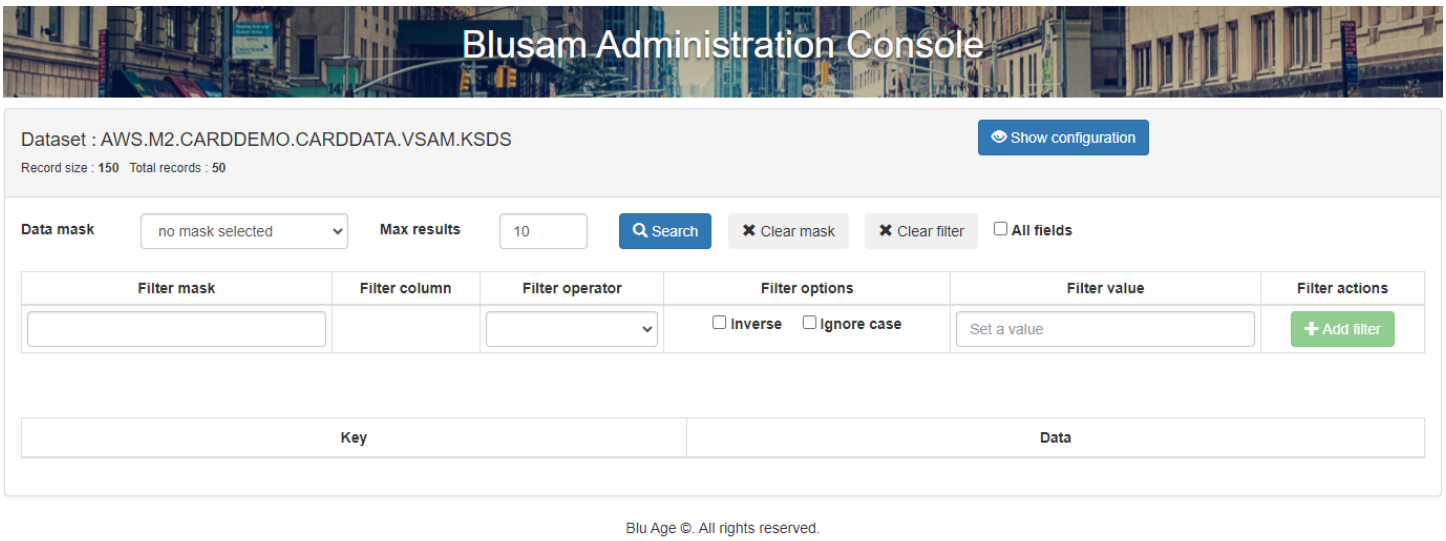
清單中的每個項目都是使用中的連結，可以對資料集執行指定的動作：

- 讀取：瀏覽資料集中的記錄
- 載入：從舊式資料集檔案匯入記錄
- 匯出：將記錄匯出為平面檔案 (與舊版系統相容)
- 清除：移除資料集中的所有記錄
- 刪除：從儲存裝置中移除資料集

下列各節提供每個動作的詳細資訊。

瀏覽資料集中的記錄

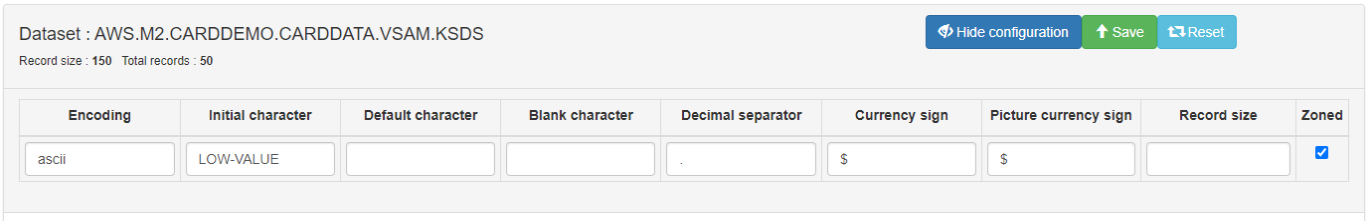
當您針對指定的資料集選擇「讀取」動作時，會出現下列頁面。



該頁面是由：

- 一個標題，具有：

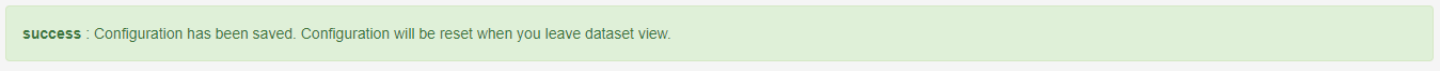
- 資料集：資料集名稱
- 記錄大小：固定記錄長度，以字節表示
- 記錄總計：此資料集儲存的記錄總數
- 顯示組態按鈕 (位於右側)：顯示/隱藏資料集組態的切換按鈕。首先，配置是隱藏的。使用按鈕時，配置會看到配置，如下圖所示。



  顯示配置時，兩個新按鈕：「儲存」和「重設」，分別用於：

  - 儲存此資料集和目前工作階段的組態
  - 將所有欄位的組態重設為預設值。
- 可設定屬性的清單，可針對指定的資料集量身打造瀏覽體驗。

可配置屬性與中描述的組態屬性相符the section called “BAC專用配置文件”。請參閱該章節以瞭解每個資料行的意義和適用值。您可以在此處為資料集重新定義每個值，並為工作階段儲存 (使用「儲存」(Save) 按鈕)。儲存組態之後，會出現類似下圖所示的橫幅。



標題會指出當您離開目前頁面時，工作階段會結束。

配置部分中沒有記錄一個額外的可配置屬性：記錄大小。這是用來指定給定的記錄大小（以字節表示），以字節表示，該記錄將過濾此數據集的適用掩碼：只有總長度與給定記錄大小匹配的掩碼才會列在「數據掩碼」下拉列表中。

使用附近的所有選項和篩選器，從資料集擷取記錄會由「搜尋」按鈕觸發。

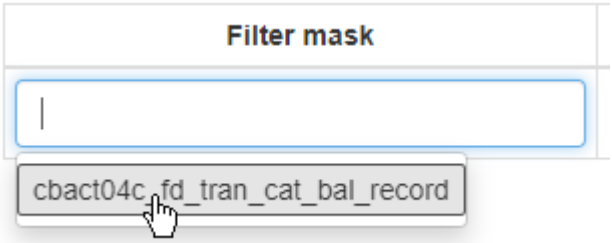
選項的第一行：

- 資料遮罩下拉式清單會顯示適用的遮罩 (根據記錄大小)。請注意，匹配記錄大小不足以成為有效的適用掩碼。遮罩定義也必須與記錄內容相容。這裡選擇的數據掩碼有
- 結果上限：限制搜尋擷取的記錄數。如果是資料集中的無限制 (分頁) 結果，則設定為 0。
- 搜尋按鈕：使用篩選條件和選項啟動記錄擷取

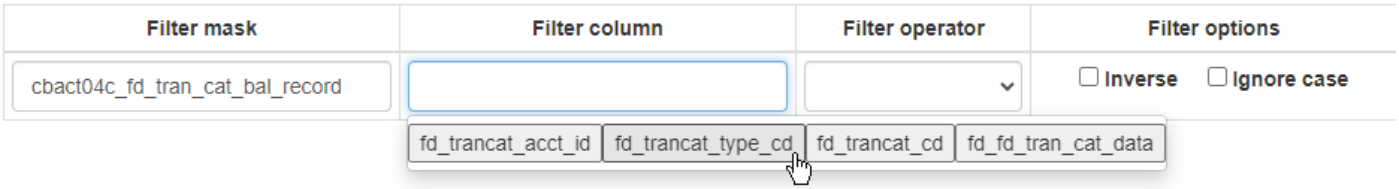
- 清除蒙版按鈕：將清除使用的遮罩（如果有），並將結果頁面切換回原始鍵/數據演示文稿。
- 清除過濾器按鈕：將清除使用的過濾器（如果有），並相應地更新結果頁面。
- 所有欄位切換：選取時，使用定義的`skip = true`遮罩項目仍會顯示，否則會隱藏遮罩項目。`skip = true`

下一行篩選條件：根據套用至指定遮罩之欄位 (欄) 的篩選條件，可以定義篩選條件清單，如下圖所示。

- 篩選遮罩：要從中挑選篩選資料行的遮罩名稱。如果您選擇該字段，那么系統會顯示相應的屏蔽列表。您可以從該列表中選擇所需的蒙版。

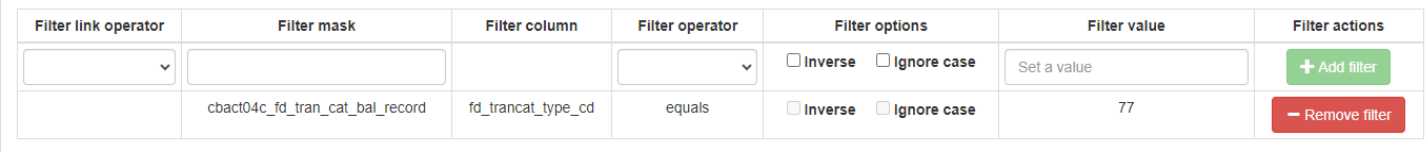


- 篩選欄：遮罩中的欄位 (欄) 名稱，用於篩選記錄。當您選擇欄位時，會顯示遮罩欄清單。若要填入「篩選」欄位，請選擇所需的儲存格。



- 篩選運算子：要套用至所選欄的運算子。以下是可用的運算子。
  - equals for：記錄的列值必須等於過濾器值
  - 開頭為：記錄的欄值必須以篩選器值開頭
  - 結尾為：記錄的列值必須以過濾器值結束
  - 包含：記錄的列值必須包含過濾器值
- 篩選選項：
  - 逆：對過濾器運算符應用逆條件; 例如，「等於」被「不等於」替換;
  - 忽略大小寫：忽略篩選運算子的字母數字比較的大小寫
- 篩選值：篩選運算子與篩選欄比較時所使用的值。

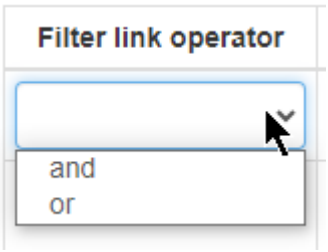
一旦設定了最少數目的篩選器項目 (至少必須設定篩選遮罩、篩選資料行、篩選運算子和篩選器值)，就會啟用 [新增篩選器] 按鈕，然後按一下它會在擷取的記錄上建立新的篩選條件。在頂部添加了另一個空的過濾條件行，並且添加的過濾條件具有「刪除過濾器」按鈕，可用於抑制給定的過濾條件：



當您啟動搜尋時，篩選的結果會顯示在分頁表格中。

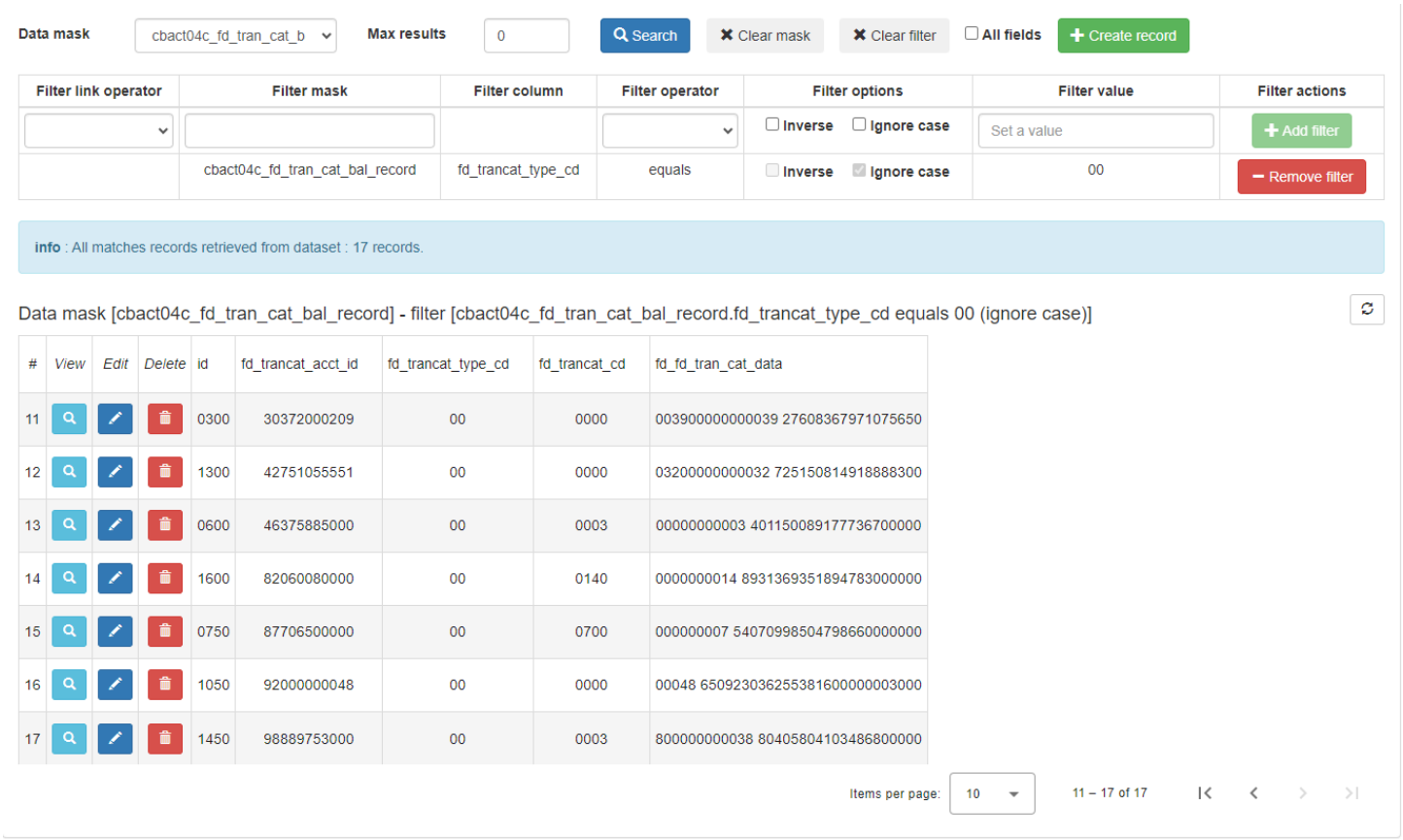
注意

- 連續篩選器由和或或連結。每個新的篩選器定義都會從設定連結運算子開始，如下圖所示。



- 可能沒有符合指定篩選條件的任何記錄。

否則，結果表看起來就像下圖中的結果表。



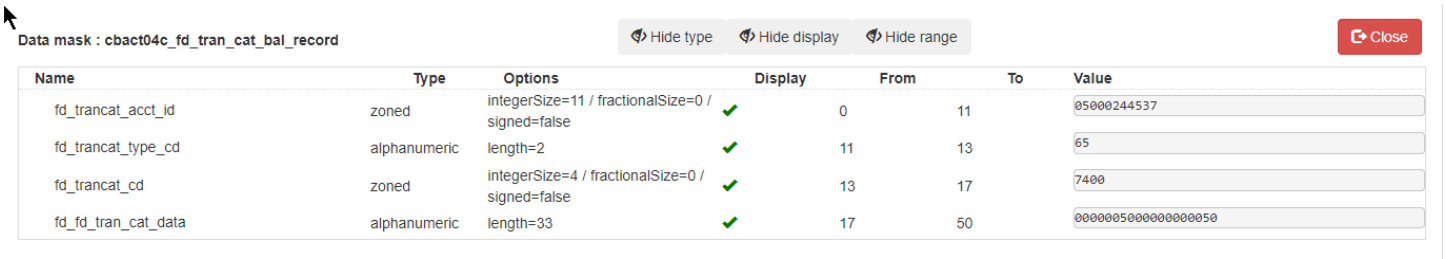
標題表示符合篩選條件的記錄總數。在標題之後，您會看到以下內容。

- 提醒使用的資料遮罩 (如果有的話) 和篩選條件。
- 一個刷新按鈕，您可以使用 Blusam 存儲中的最新值觸發整個結果表的刷新（例如，因為它可能已被另一個用戶更新）。

對於每個擷取的記錄，資料表都會有一列，顯示將資料遮罩套用至資料錄內容的結果。每一列是根據列的類型（並使用選擇的編碼）對記錄子部分的解釋。在每一行的左側，有三個按鈕：

- 放大鏡按鈕：導向顯示詳細記錄內容的專用頁面
- 畫筆按鈕：導向記錄內容的專用編輯頁面：
- 垃圾桶按鈕：用於從 blusam 存儲中刪除給定的記錄

詳細查看記錄的內容：



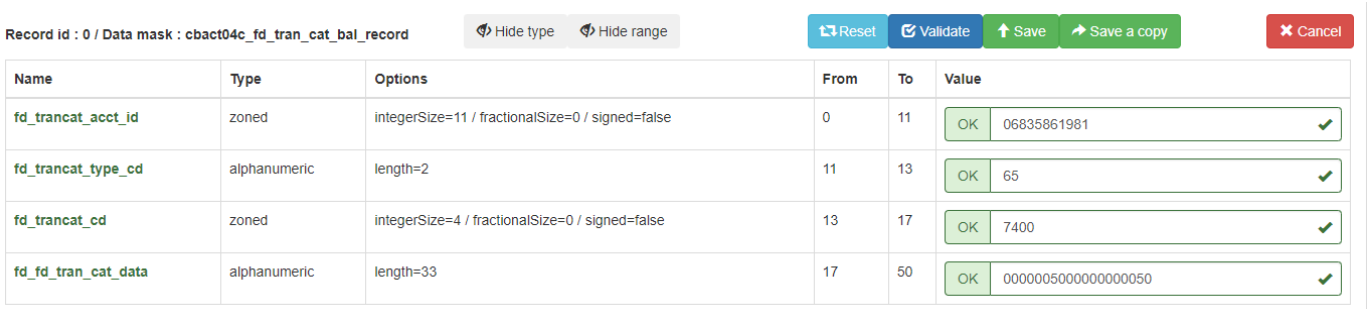
- 三切換按鈕隱藏或顯示一些列：
  - 隱藏/顯示類型
  - 隱藏/顯示顯示標誌
  - 隱藏/顯示範圍
- 若要離開此專用頁面並返回結果表格，請選擇「關閉」。
- 每一列代表資料遮罩中的一欄，其中包含下列欄：
  - 名稱：資料欄的名稱
  - 類型：資料欄的類型
  - 顯示：顯示指示器；如果使用定義符合的遮罩項目，則會顯示綠色勾號`skip = false`，否則會顯示紅色十字
  - 起始與終止：以 0 為基準的記錄子部分範圍
  - 值：記錄子部分的解釋值，使用類型和編碼

編輯記錄的內容：

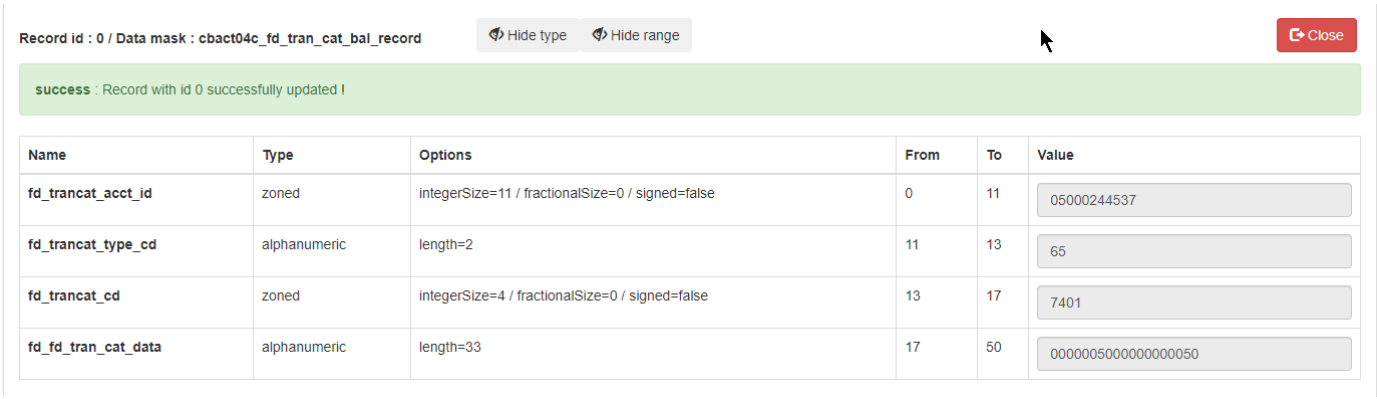
編輯頁面與上述檢視頁面類似，不同之處在於遮罩項目值是可編輯的。三個按鈕控制更新程序：

- 重置：將可編輯值重置為初始記錄值（在任何版本之前）；
- 驗證：驗證輸入，關於掩碼項目類型。對於每個遮罩項目，驗證結果將使用可視化標籤（OK如果驗證成功，則為複選框，如果驗證失敗，則為紅十字，以ERROR及提供有關驗證失敗提示的錯誤消息）來打印驗證結果。如果驗證成功，將出現兩個新按鈕：

- 保存：嘗試將現有記錄更新到 Blusam 存儲
- 保存副本：嘗試在 Blusam 存儲中創建一個新記錄



- 如果將記錄儲存成功，則會顯示一則訊息，並且頁面會切換到唯讀模式 (無法再編輯遮罩項目值)：

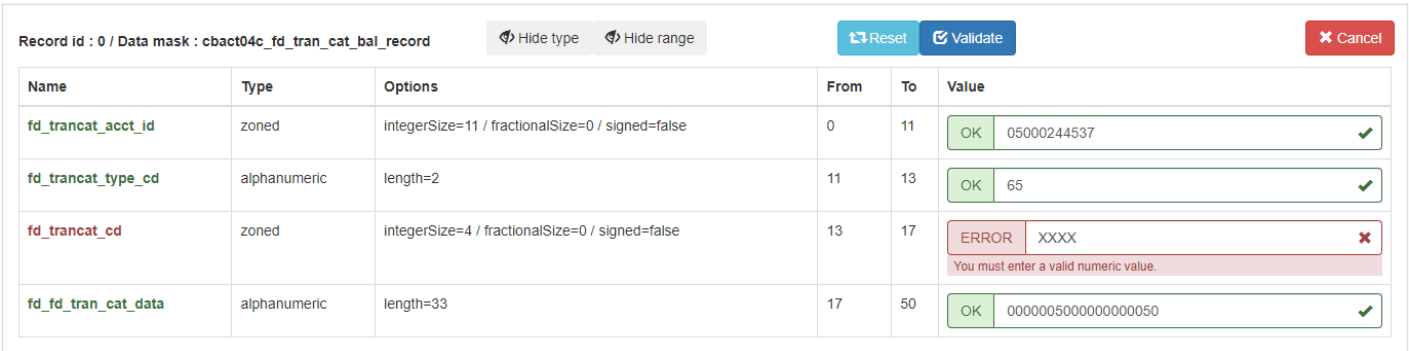


- 如果因任何原因導致記錄持續存放裝置失敗，錯誤訊息會顯示為紅色，提供失敗原因。最常見的失敗情況是存儲記錄會導致密鑰損壞（無效或重複的密鑰）。如需插圖，請參閱下列註記。
- 若要結束，請選擇「關閉」按鈕。

- 取消：結束編輯階段作業，關閉頁面，然後返回記錄清單頁面。

請注意：

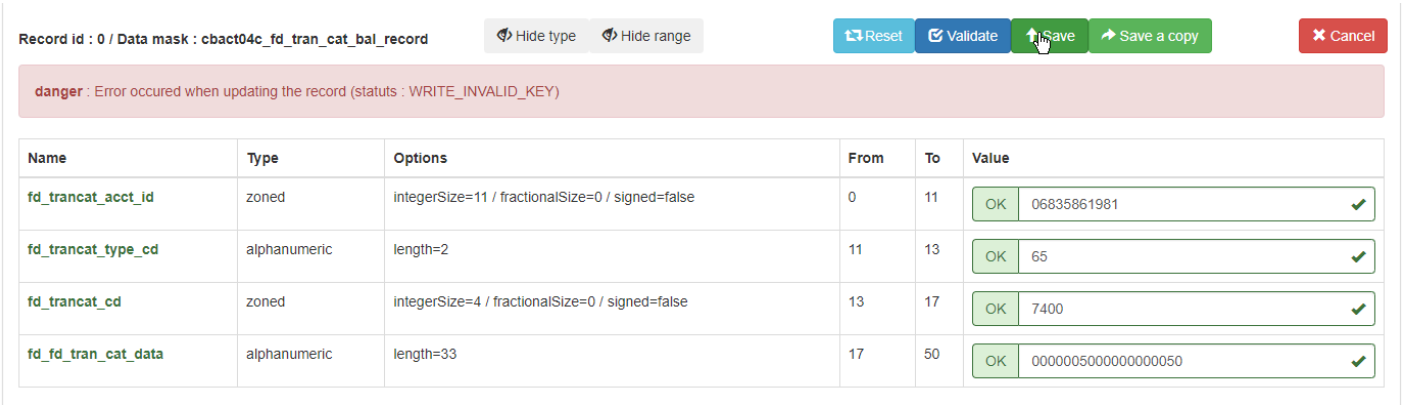
- 驗證機制只會檢查遮罩項目值是否與遮罩項目類型正式相容。例如，請參閱數字遮罩項目上的此驗證失敗：



- 驗證機制可能會嘗試自動更正無效輸入，並以藍色顯示資訊訊息，表示該值已根據其類型自動更正。例如，輸入 7XX0 作為數值遮罩項目中的數fd_trncat_cd值：

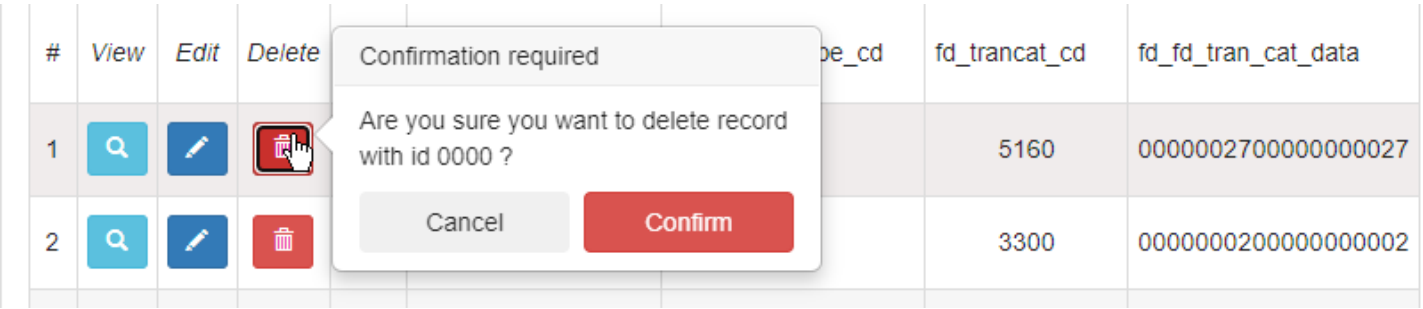
呼叫驗證會導致下列情況：

- 驗證機制不會檢查給定值是否在密鑰完整性方面是否有效（如果給定的數據集涉及任何唯一密鑰）。例如，儘管驗證成功，如果提供的值導致無效或重複的密鑰情況，持久性將失敗，並顯示錯誤消息：



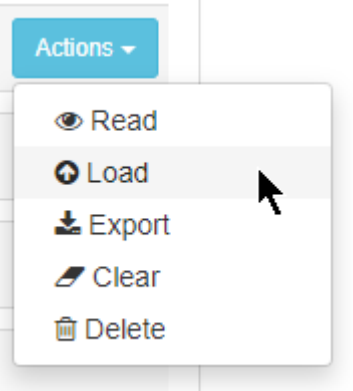
刪除記錄：

若要刪除記錄，請選擇 [垃圾桶] 按鈕：

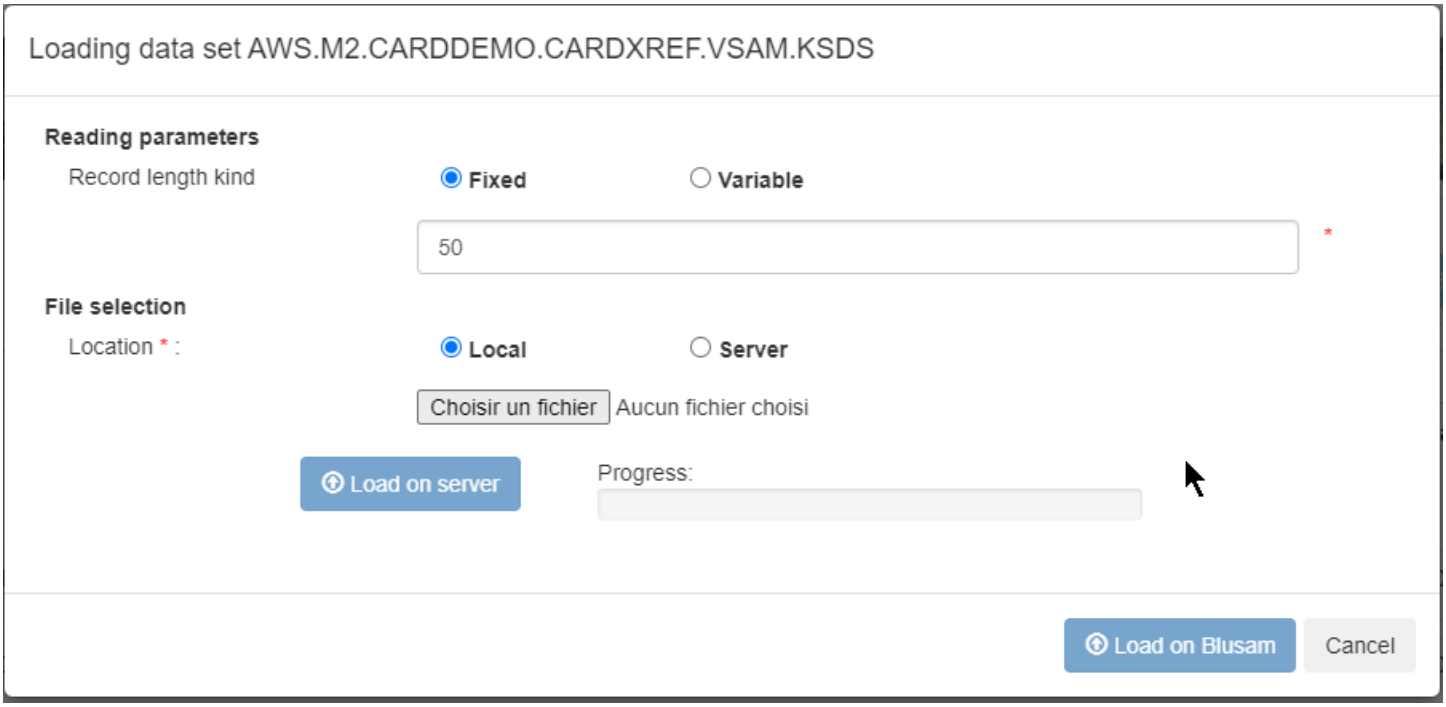


將記錄載入資料集

若要將記錄載入資料集，請選擇「動作」，然後選擇「載入」。



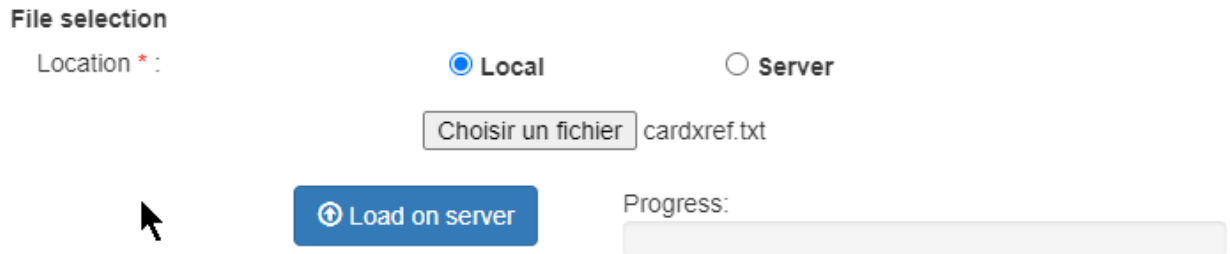
隨即顯示含有載入選項的視窗。



首先，「在伺服器上載入」和「在 Blusam 上載入」按鈕都會停用。

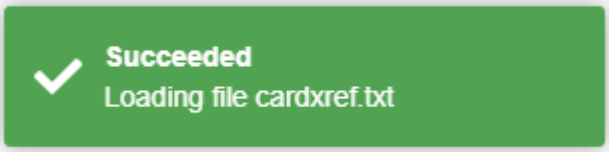
讀取參數：

- 記錄長度種類：
  - 固定或可變記錄長度：使用單選按鈕來指定舊式資料集匯出是使用固定長度記錄還是可變長度記錄(記錄應以RDW位元組開頭)。如果您選擇「固定」(Fixed)，則必須在輸入欄位中將記錄長度指定為正整數值 (以位元組為單位)。該值應由來自數據集的信息預先填充。如果你選擇變量，給定的輸入字段消失。
  - 檔案選擇：
    - 本地：使用下面的文件選擇器從本地計算機中選擇數據集文件（注意：文件選擇器使用瀏覽器的語言環境來打印其消息--這是法語，但在您身邊看起來可能會有所不同，這是預期的）。進行選取之後，視窗會更新為資料檔案名稱，並啟用「載入伺服器」按鈕：

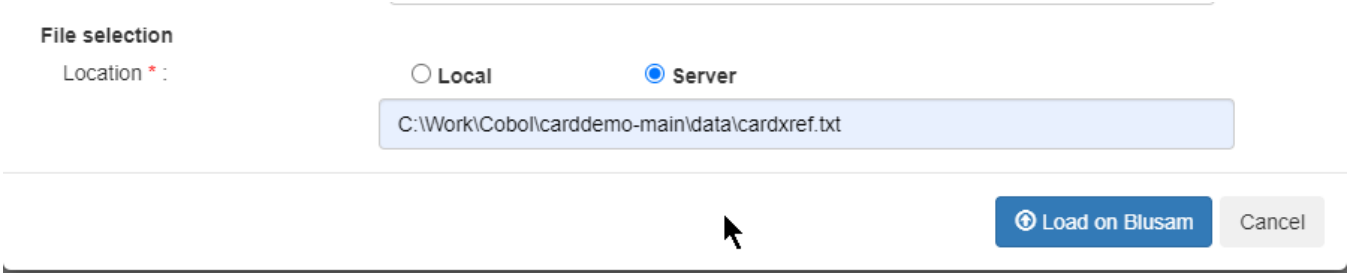


選擇在伺服器上載入。進度條到達其結束後，在 Blusam 上加載按鈕被啟用：

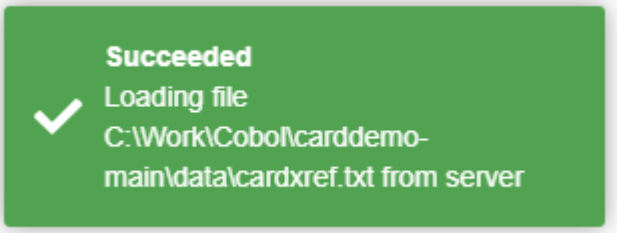
若要完成 Blusam 儲存裝置的載入程序，請選擇「在 Blusam 上載入」。否則，請選擇 Cancel (取消)。如果您選擇繼續加載過程，加載過程完成後，右下角將顯示一條通知：



- 伺服器：選擇此選項會在「載入伺服器」按鈕消失時顯示輸入欄位。您必須在輸入欄位中指定 Blusam 伺服器上資料集檔案的路徑（假設您已先將指定檔案傳輸到 Blusam 伺服器）。指定路徑後，「在 Blusam 上載入」會啟用：

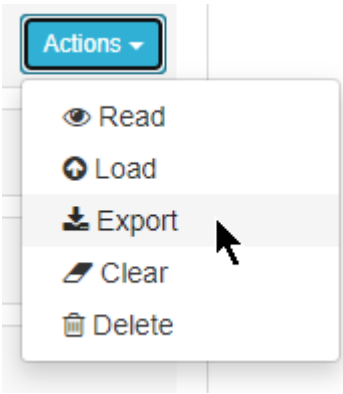


若要完成載入程序，請選擇「在 Blusam 上載入」。否則，請選擇 Cancel (取消)。如果您選擇繼續載入，載入程序完成後會顯示通知。該通知與瀏覽器的負載不同，因為它顯示數據文件服務器路徑，後跟來自服務器的單詞：

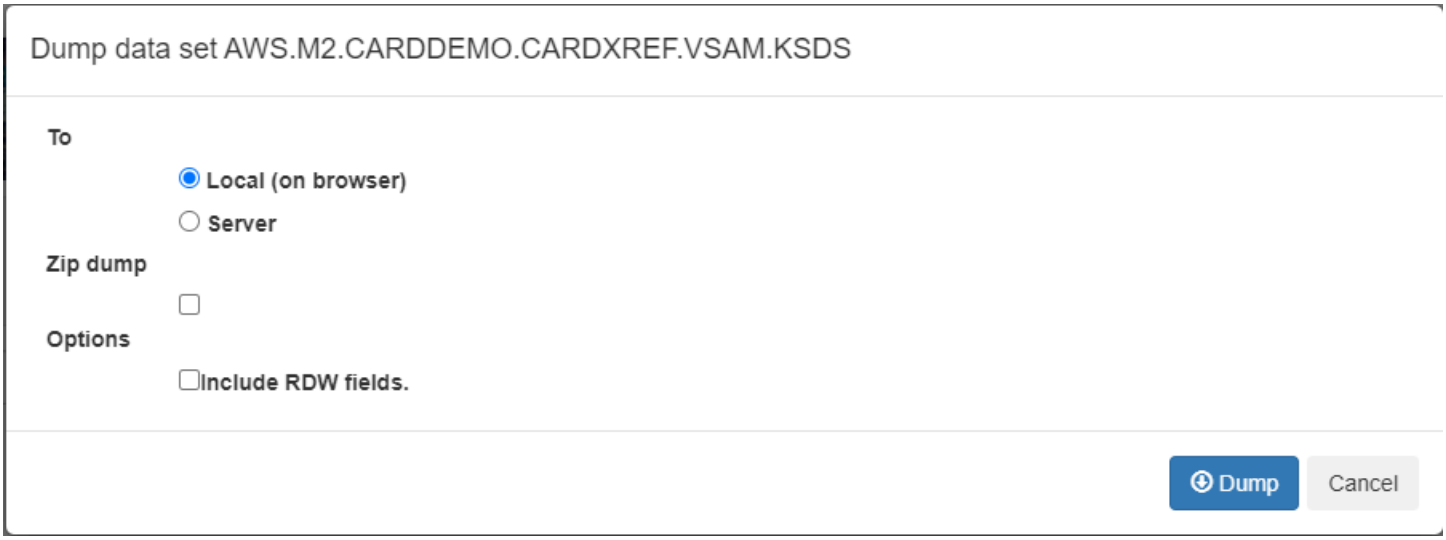


從資料集匯出記錄

若要匯出資料集記錄，請選擇目前資料集列中的「動作」，然後選擇「匯出」：

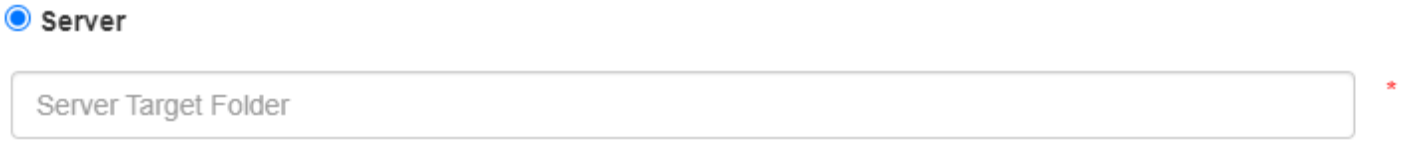


會出現下列快顯視窗。



選項：

至：單選按鈕選擇，用於選擇導出目的地，可以作為瀏覽器中的下載（本地（在瀏覽器上））或託管BAC應用程序的服務器上的給定文件夾。如果您選擇使用「伺服器」選項匯出，則會顯示新的輸入欄位：



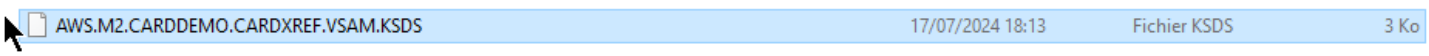
正如輸入字段右側的紅色星號所示，必須在服務器上提供有效的文件夾位置（「轉儲」按鈕將處於非活動狀態，而沒有提供任何文件夾位置）。

若要匯出至伺服器，如果您計劃在匯出之後操作匯出的資料集檔案，則必須具備伺服器檔案系統的足夠存取權限。

Zip 轉儲：生成壓縮存檔而不是原始文件的複選框。

選項：若要在變數長度記錄資料集的情況下，在匯出的資料集中每筆記錄的開頭包含「記錄描述元字」(RDW)，請選擇「包含RDW欄位」。

若要啟動資料集匯出程序，請選擇「傾印」。如果您選擇匯出至瀏覽器，請檢查匯出資料集檔案的下載資料夾。該文件將具有與數據集相同的名稱：

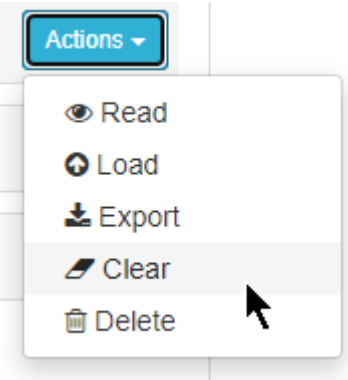


請注意：

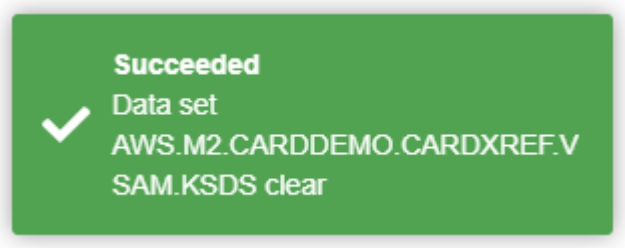
- 對於KSDS，記錄將按照主鍵順序導出。
- 對於ESDS和RRDS，記錄將按照RBA（相對字節地址）順序導出。
- 對於所有數據集類型，記錄將導出為原始二進制數組（不會發生任何類型的轉換），從而確保與舊版平台的直接兼容。

清除資料集中的記錄

若要清除資料集中的所有記錄，請選擇「作業」，然後選擇「清除」：

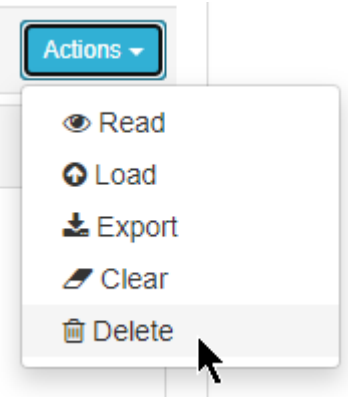


從資料集中移除所有記錄後，會出現下列通知。

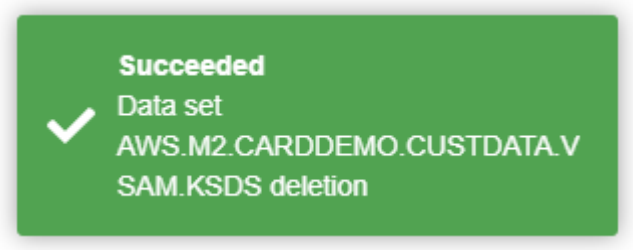


刪除資料集

若要刪除資料集，請選擇「動作」，然後選擇「刪除」：



刪除資料集後，會出現下列通知：

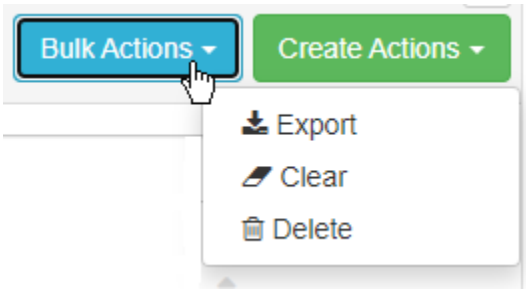


批量操作

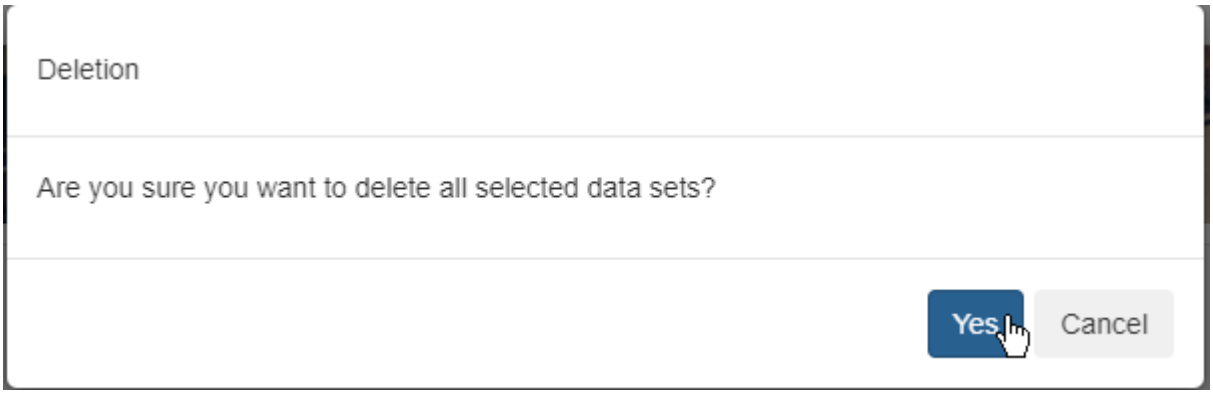
資料集上有三個大量作業可用：

- 匯出
- Clear
- Delete

批次處理作業只能套用至選取的資料集 (至少需要選取一個資料集)；選取資料集是透過在資料集清單表格中資料集列左側的勾選核取方塊來完成資料集。選取至少一個資料集將啟用「批次處理動作」下拉式清單：



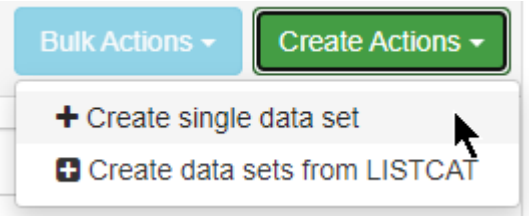
除了給定的操作適用於選擇的數據集而不是單個數據集的事實之外，這些操作與上述操作類似，因此請參閱專用操作文檔以獲取詳細信息。彈出窗口的文本內容將略有不同，以反映批量性質。例如，當嘗試刪除多個數據集時，彈出窗口將如下所示：



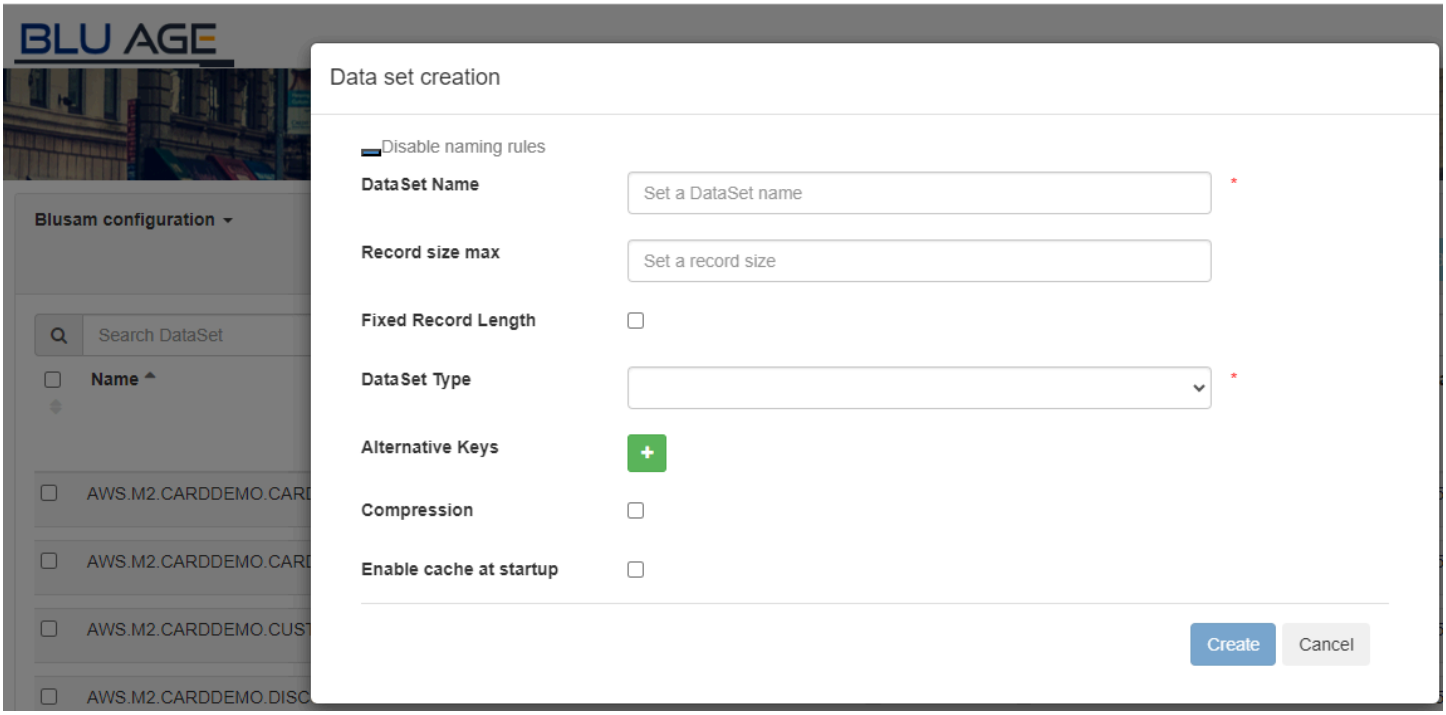
建立作業

建立單一資料集

選擇「動作」，然後選擇「建立單一資料集」：

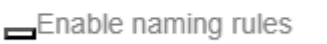


然後，資料集建立表單將顯示為快顯視窗：

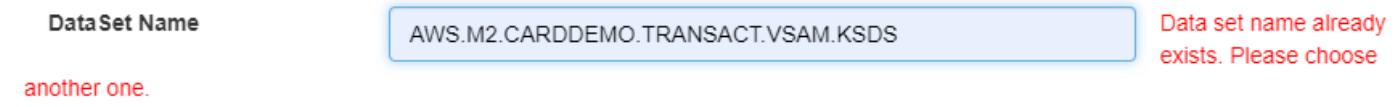


您可以為資料集定義指定下列屬性：

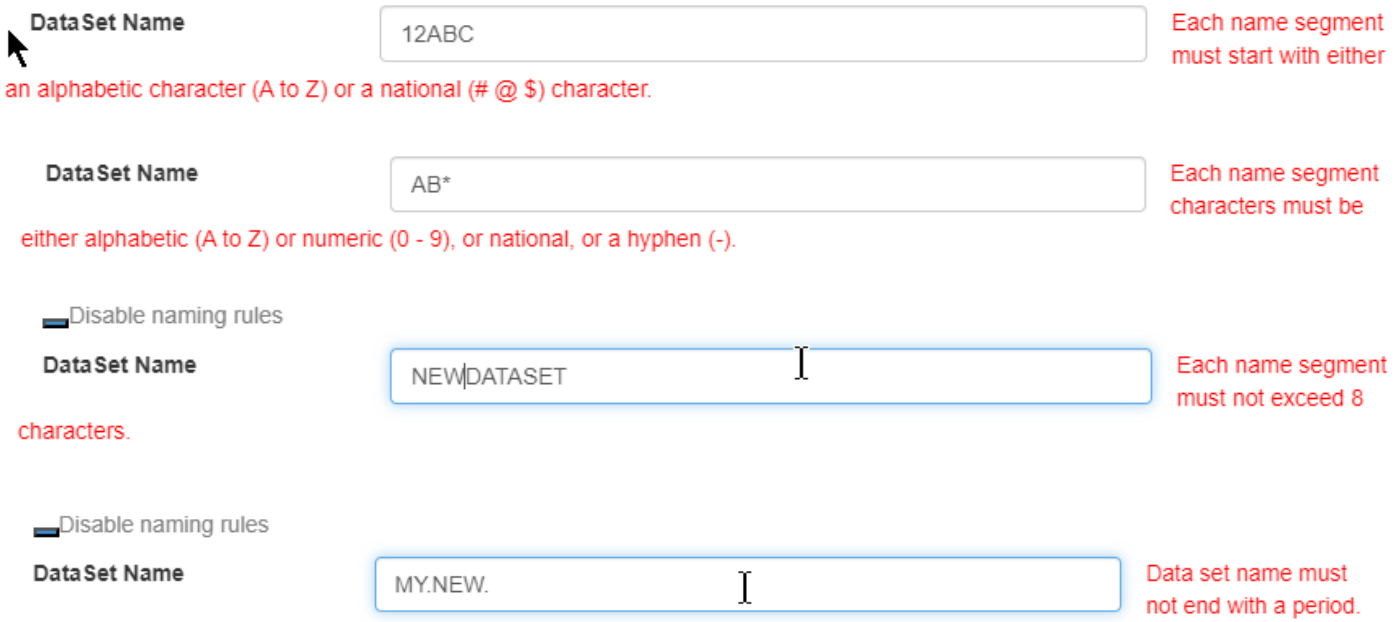
- 啟用和停用命名規則：使用「停用命名規則/啟用命名規則」切換 Widget 來停用和啟用資料集命名慣例。我們建議您將開關保留為預設值，並啟用資料集命名規則 (切換 Widget 應顯示「停用命名規則」)：



- 資料集名稱：資料集的名稱。如果您指定的名稱已在使用中，則會出現下列錯誤訊息。



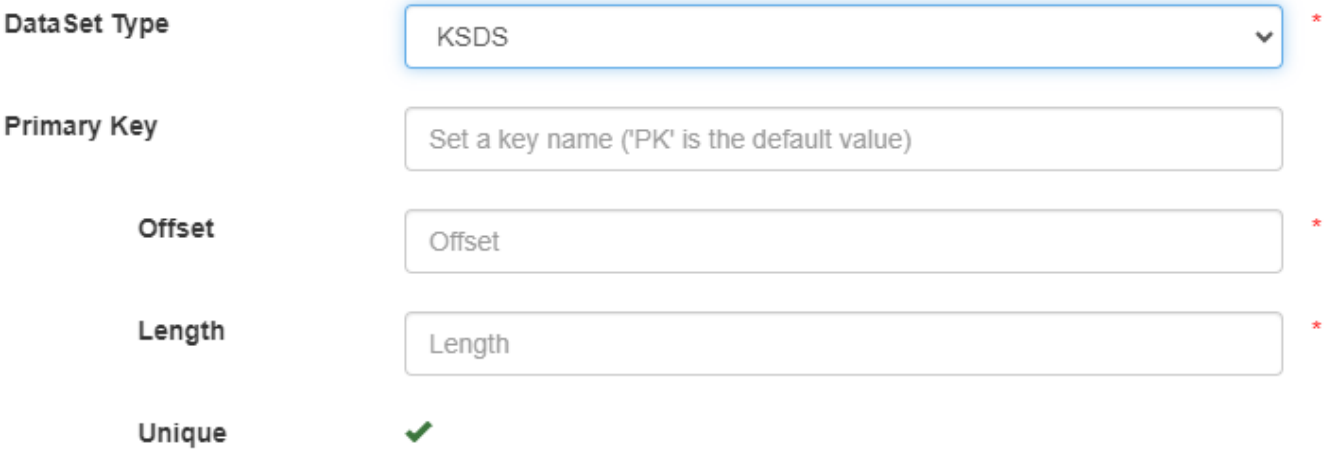
  如果已啟用，名稱也必須遵守命名慣例：



- 記錄大小 max：這必須是正整數，代表具有固定長度記錄之資料集的記錄大小。對於具有可變長度記錄的資料集，您可以將其保留為空白。
- 固定長度記錄：用於指定記錄長度是固定的還是可變的核取方塊。如果選取此選項，資料集將具有固定長度的記錄，否則記錄長度將會變數。

  當您將舊資料匯入變數長度記錄資料集時，提供的舊記錄必須包含記錄描述元字 (RDW)，以提供每筆記錄的長度。
- 資料集類型：用於指定目前資料集類型的下拉式清單。支援下列類型。
  - ESDS
  - 大 ESDS
  - KSDS

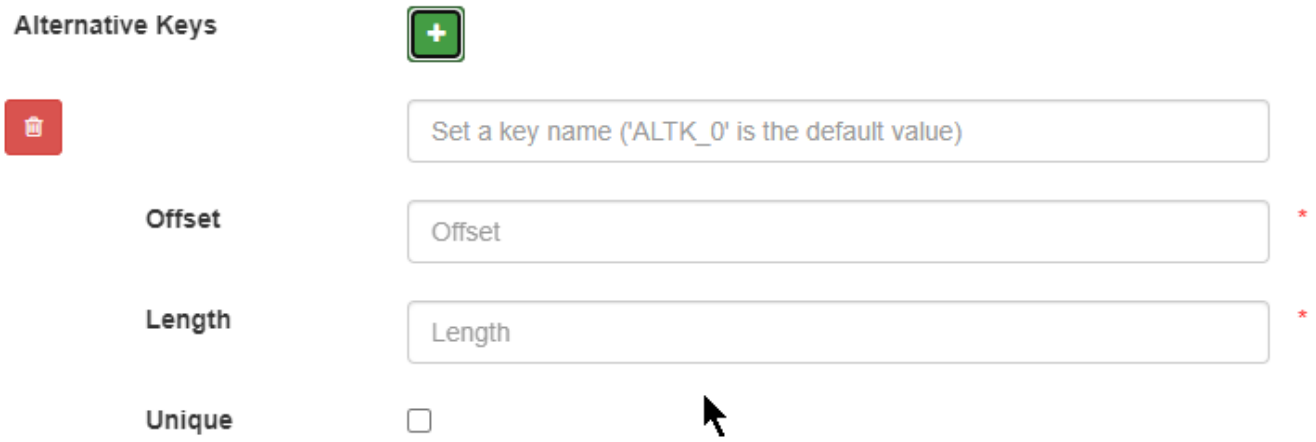
  對於KSDS，您必須指定主索引鍵：



對於主索引鍵，請指定下列項目：

- 名稱：此欄位為選擇性欄位。預設值為 **PK**。
- 偏移量：記錄內主鍵的基於 0 的偏移量。偏移量必須是正整數。此欄位為必填。
- 長度：主鍵的長度。此長度必須為正整數。此欄位為必填。

對於KSDS和ESDS，您可以選擇性地定義替代索引鍵的集合，方法是選擇性地選擇性地選擇性地選擇性地選擇性地選擇性地選擇性地 每次選擇該按鈕時，資料集建立表單中都會顯示新的替代索引鍵定義區段：



對於每個替代鍵，您需要提供：

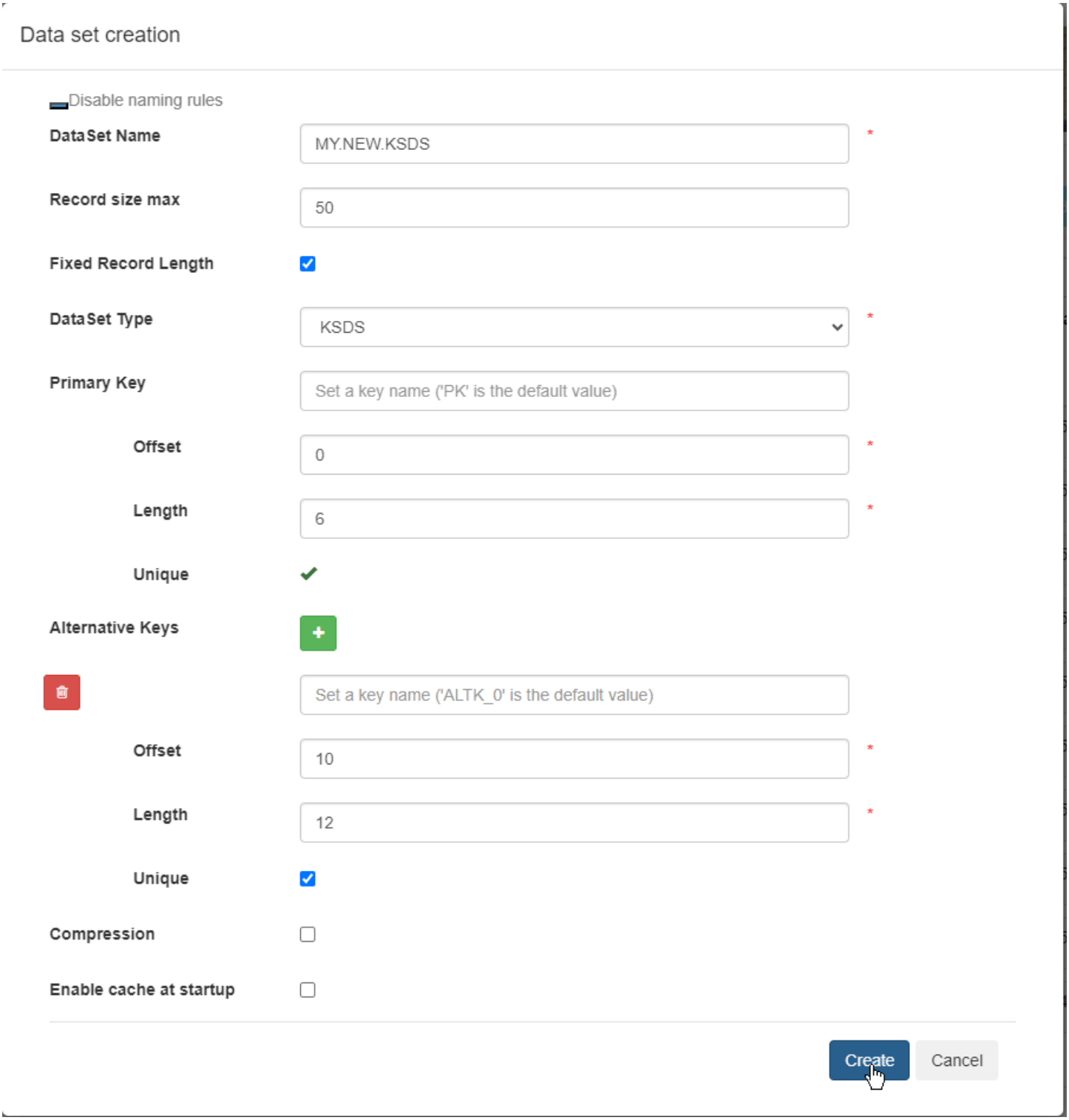
- 名稱：此欄位為選擇性欄位。預設值為**ALTK_#**，其中 # 代表從 0 開始的自動遞增計數器。
- 偏移：記錄內替代鍵的 0 基於偏移。必須是正整數。此欄位為必填。
- 長度：替代鍵的長度。此長度必須為正整數。此欄位為必填。

- 唯一：指示替代鍵是否接受重複項目的核取方塊。如果選取此選項，則替代鍵會定義為唯一 (NOT 接受重複的按鍵項目)。此欄位為必填。

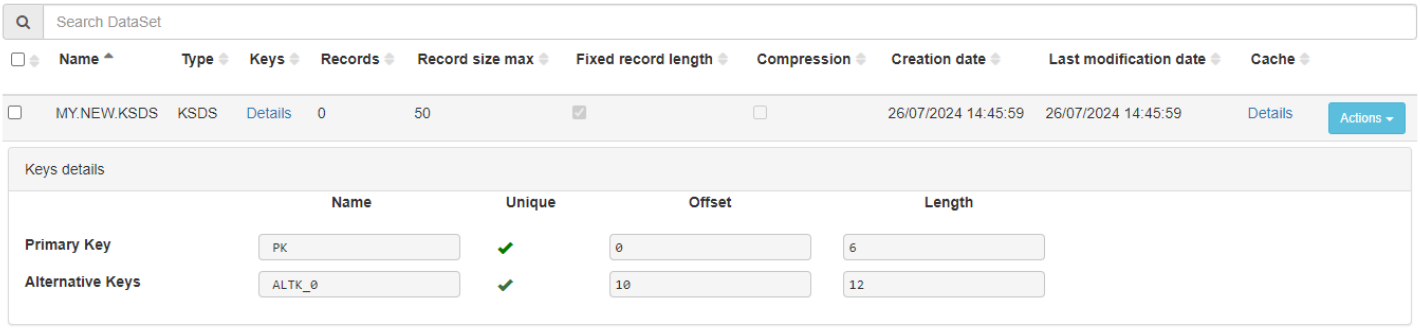
若要移除備用金鑰定義，請使用左側的垃圾桶按鈕。

- 壓縮：指定是否要使用壓縮來儲存資料集的核取方塊。
- 啟動時啟用快取：用來指定是否應在應用程式啟動時將資料集載入快取的核取方塊。

指定屬性定義之後，請選擇建立以繼續：



建立視窗將會關閉，並顯示顯示資料集清單的首頁。您可以檢視新建立的資料集的詳細資訊。



建立資料集 LISTCAT

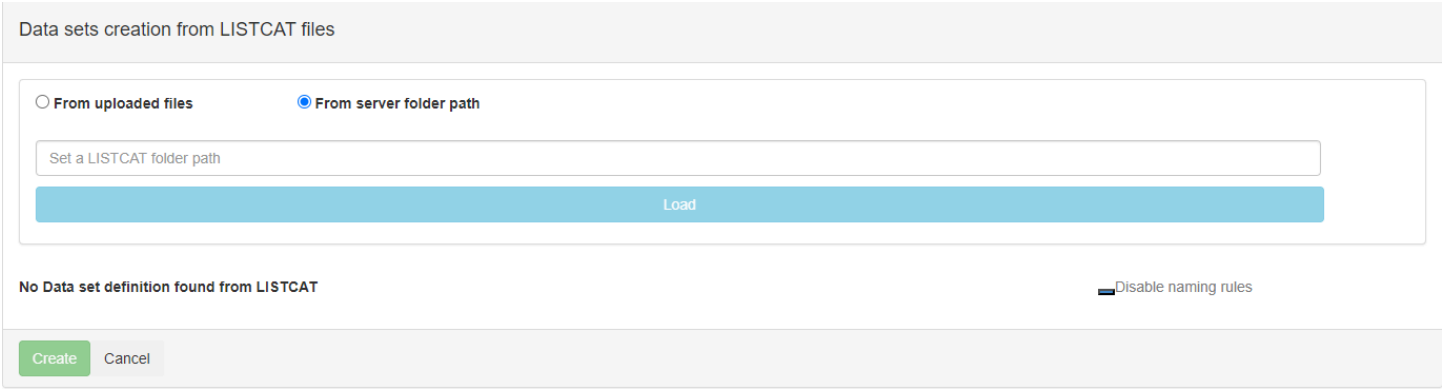
此功能可讓您利用 BluAge 轉換過程中使用 Transable Center 建立的LISTCATJSON檔案，作為從舊式平台剖析LISTCAT匯出的結果：LISTCAT匯出會剖析並轉 BluInsights 換成保存資料集定義的JSON檔案 (名稱、資料集類型、金鑰定義，以及記錄長度是固定或變數)。

擁有這些LISTCATJSON文件可以直接創建數據集，而無需手動輸入數據集所需的所有信息。您也可以直接建立資料集合，而不必逐一建立資料集合。

如果您的計划沒有可用的LISTCATJON文件（例如，因為轉換時沒有可用的LISTCAT導出文件），那么您可以手動創建一個文件，前提是您必須遵循附錄中詳述的LISTCATJSON格式。

從「建立動作」下拉式清單中，選擇「從中建立資料集」LISTCAT。

將顯示以下專用頁面：



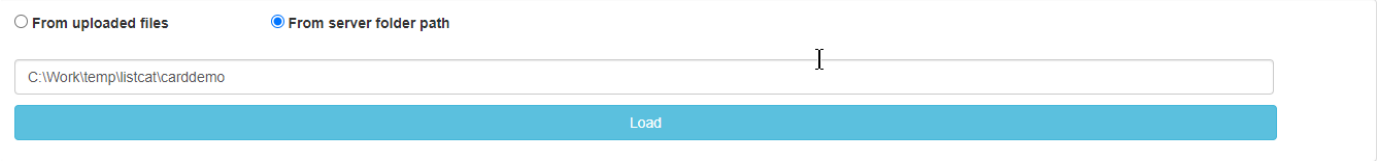
在這個階段，加載按鈕被禁用，這是預期的。

使用選項按鈕指定提供LISTCATJSON檔案的方式。有兩個選項：

- 您可以使用瀏覽器上傳JSON文件。
- 您可以從伺服器上的資料夾位置選取JSON檔案。若要選擇此選項，您必須先將JSON檔案複製到具有適當存取權限的伺服器上的指定資料夾路徑。

使用伺服器上的JSON檔案

1. 設置服務器上的文件夾路徑，指向包含LISTCATJSON文件的文件夾：

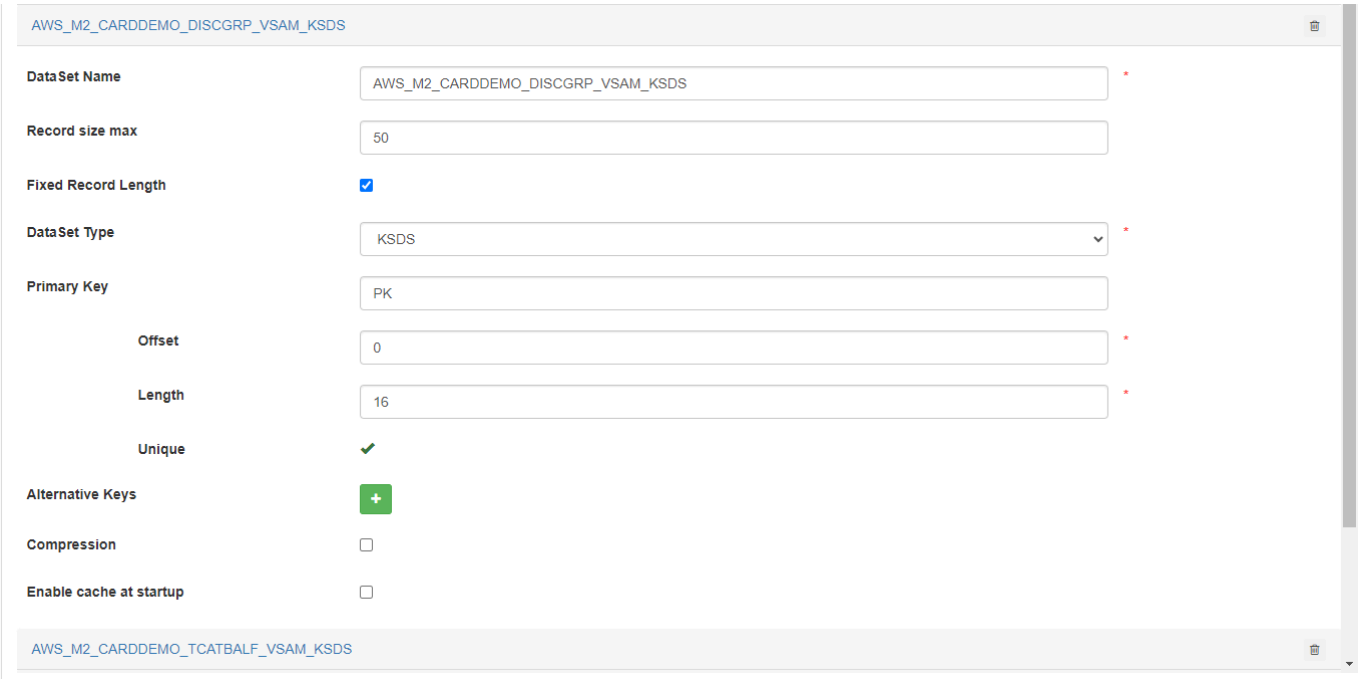


2. 選擇「載入」按鈕。所有可辨識的資料集定義都會列在表格中：

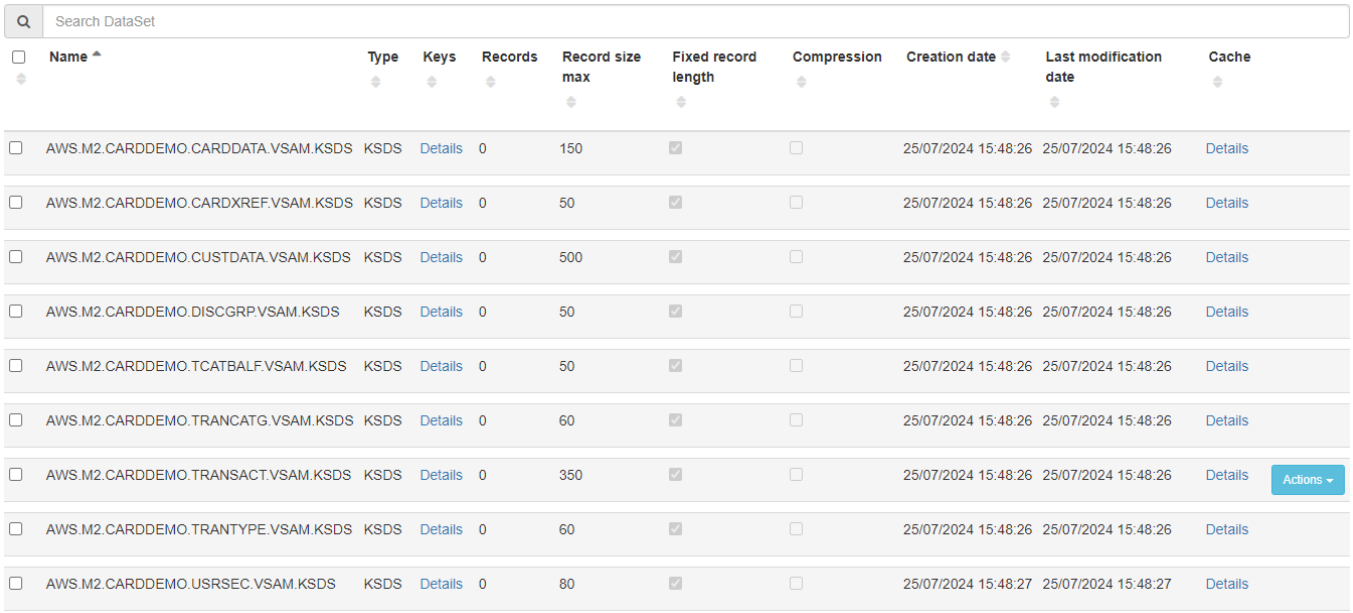
每一列代表一個資料集定義。您可以使用垃圾桶按鈕從清單中移除資料集定義。

> **Important**
>
> 從列表中刪除是立即的，沒有警告消息。

3. 左邊的名字是一個鏈接。您可以選擇它來顯示或隱藏可編輯的資料集定義的詳細資訊。您可以自由修改定義，從解析的JSON文件的基礎上開始。

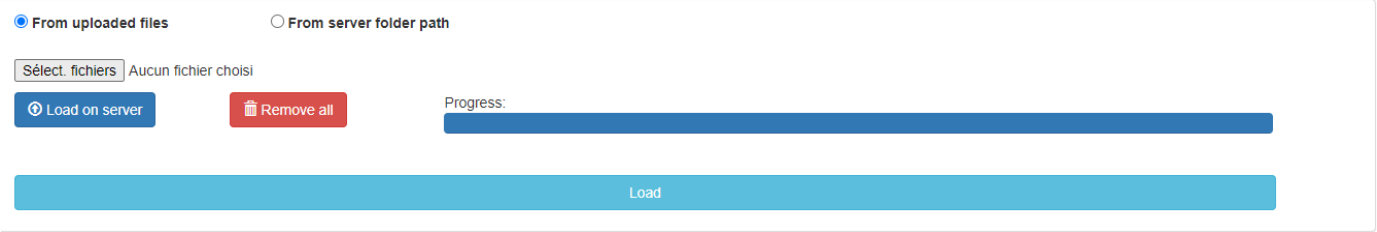


4. 若要建立所有資料集，請選擇 [建立]。將建立所有資料集，並將顯示在資料集結果頁面上。新創建的數據集將全部有 0 條記錄。



將檔案上傳至伺服器

1. 此選項類似於使用伺服器資料夾路徑中的檔案，但在此情況下，您必須先使用檔案選擇器上傳檔案。選取要從本機電腦上傳的所有檔案，然後選擇「在伺服器上載入」。



2. 當進度列到達結尾時，所有檔案都已成功上傳到伺服器，並啟用「載入」按鈕。選擇「載入」按鈕，並使用發現的資料集定義，如先前所述。

LISTCATJSON格式

格LISTCATJSON式由下列屬性定義：

- 可選 "catalogId「: 舊式目錄的識別碼 (做為字串)，或預設目錄的「預設」識別碼。
- 「標識符」：數據集名稱，作為一個字符串。
- 「isIndexed「：一個布爾標誌來表示KSDS：真為KSDS，否則為 false。
- 「isLinear「：一個布爾標誌來表示ESDS：真為ESDS，否則為 false。
- 「isRelative「：一個布爾標誌來表示RRDS：真為RRDS，否則為 false
- 注意：isIndexed""、isLinear "" 和 isRelative "是互斥的。
- 「isFixedLength記錄」：布爾標誌：如果固定長度記錄數據集，則設置為 true，否則設置為 false。
- 「avgRecordSize「：以字節為單位的平均記錄大小，以正整數表示。
- 「maxRecordSize「：以字節為單位的最大記錄大小，以整數表示。固定長度記錄大小應等於。avgRecordSize
- KSDS僅用於：強制性主鍵定義（作為嵌套對象）
  - 標籤為「」primaryKey
  - 「offset」：記錄中主鍵的基於 0 的字節偏移量。
  - 「長度」：主鍵的長度（以字節為單位）。
  - 「唯一」：主鍵必須設置為 true。
- forKSDS/ESDS，備用鍵的集合（作為嵌套對象的集合）：
  - 標籤為「」alternateKeys
  - 對於每個備用鍵：
    - 「offset」：記錄中備用鍵的基於 0 的字節偏移量。
    - 「length」：備用索引鍵的長度 (以位元組為單位)。

    - 「唯一」：必須為備用鍵設置為 true，如果鍵不接受重複的條目，否則為 false。
- 如果沒有備用索引鍵存在，請提供空集合：

```
alternateKeys: []
```

以下是範例KSDSLISTCATJSON檔案。

```
{
  "catalogId": "default",
  "identifier": "AWS_M2_CARDDEMO_CARDXREF_VSAM_KSDS",
  "isIndexed": true,
  "isLinear": false,
  "isRelative": false,
  "isFixedLengthRecord": true,
  "avgRecordSize": 50,
  "maxRecordSize": 50,
  "primaryKey": {
    "offset": 0,
    "length": 16,
    "unique": true
  },
  "alternateKeys": [
    {
      "offset": 25,
      "length": 11,
      "unique": false
    }
  ]
}
```

為 AWS 藍光時代運行時設置配置

AWS 藍光時代運行時和客戶端代碼是使用 S pring Boot 框架的 Web 應用程序。它利用 Spring 功能來提供配置，並提供數個可能的位置和優先級規則。提供許多其他文件也有類似的優先級規則，例如 groovy 腳本，sql 等。

AWS Blu Age 運行時還包含其他可選的 Web 應用程序，可以根據需要選擇加入。

主題

- 應用程式組態基
- 應用優先權
- JNDI用於資料庫
- AWS 藍光時代運行時秘密
- 其他文件（常規，SQL 等）
- 其他 Web 應用程式
- 啟用 AWS 藍光時代運行時的屬性
- AWS 藍光時代運行時中可用的 Redis 緩存屬性
- 設定 Gapwalk 應用程式的安全性

應用程式組態基

處理應用程序配置的默認方法是通過使用應用程序服務器YAML文件config夾中提供的專用文件。有兩個主要的YAML配置文件：

- `application-main.yaml`
- `application-`*`profile`*`.yaml`(其中*`profile`*值是在應用程式產生期間設定的)。

第一個文件配置框架，即`Gapwalk-application.war`，而第二個文件專門用於客戶端應用程序的其他選項。這適用於彈簧輪廓的使用：Gapwalk 應用程式會使用設定`main`檔，而用戶端應用程式則使用*`profile`*設定檔。

下面的例子顯示了一個典型的主YAML文件。

```
##########################################
##### JICS datasource configuration #####
##########################################
datasource:
  jicsDs:
    driver-class-name : org.postgresql.Driver
    url: jdbc:postgresql://localhost/jics
    username: jics
    password: jics
    type : org.postgresql.ds.PGSimpleDataSource

#####################################################
##### Embedded Bluesam datasource configuration #####
#####################################################
  bluesamDs :
    driver-class-name : org.postgresql.Driver
    url : jdbc:postgresql://localhost/bluesam
    username : bluesam
    password : bluesam
    type : org.postgresql.ds.PGSimpleDataSource

##########################################
##### Embedded Bluesam configuration #####
##########################################
bluesam :
  remote : false
  cache : ehcache
  persistence : pgsql #pgsql, mssql, xodus...
  ehcache:
    resource-pool:
      size: 4GB
  write-behind:
```

下列範例顯示典型的用戶端YAML檔案。

```
# Logback context logger integration.
logging.config : classpath:logback-[redacted].xml
# Limits Spring logger output.
logging.level.org.springframework.beans.factory.support.DefaultListableBeanFactory : WARN
logging.level.org.springframework.statemachine : WARN
# If the datasource support mode is not static-xa, spring JTA transactions autoconfiguration must me disabled
spring.jta.enabled : false

spring:
  aws:
    client:
      datasources:
        names: primary
        primary:
          secret: arn:aws:secretsmanager:[redacted]

spring.jta.atomikos.datasource.primary.unique-resource-name: primary
spring.jta.atomikos.datasource.primary.xa-data-source-class-name: org.postgresql.xa.PGXADataSource
spring.jta.atomikos.datasource.primary.maxPoolSize: 20
spring.jta.atomikos.datasource.primary.autoCommit: false
```

若要取得有關YAML檔案內容的資訊，請參閱啟用 AWS 藍光時代運行時的屬性。

應用優先權

對於這些配置文件，Spring 優先級規則適用。值得注意的是：

- 該application-mainYAML文件以默認值顯示在 Gapwalk 主要 war 文件中，並且該文件config夾中的文件將取代它。
- 客戶端應用程序配置也應該完成相同的操作
- 伺服器啟動時，可以在命令列上傳遞其他參數。他們會覆蓋YAML那些。

有關更多信息，請參閱官方 Spring Boot 文檔。

JNDI用於資料庫

資料庫組態可能會在 Tomcat JNDI 中的 context.xml 檔案中隨附。任何這樣的配置都會覆蓋YAML一個。但請注意，使用此功能將不允許將您的憑據包裝在秘密管理器中（見下文）。

下列範例顯示JICS和 BluSam 資料庫的範例組態。

```
<Resource auth="Container" driverClassName="org.postgresql.Driver" initialSize="0"
 maxIdle="5"
   maxOpenPreparedStatements="-1" maxTotal="10" maxWaitMillis="-1" name="jdbc/jics"
   poolPreparedStatements="true" testOnBorrow="false" type="javax.sql.DataSource"
   url="jdbc:postgresql://XXXX.rds.amazonaws.com:5432/XXXX" username="XXXX"
 password="XXXX" />
```

jdbc /吉力

將jdbc/jics用於數JICS據庫和jdbc/bluesam（注意 'e'）為 blusam 數據庫。

網址 = "jdbc: 後盾://. 亞馬遜. COM: 5432 /" 用戶名 =」「密碼 =」」XXXX XXXX XXXX XXXX

數據庫 URL，用戶名和密碼。

AWS 藍光時代運行時秘密

某些包含認證的資源配置可以使用 AWS 密碼進一步保護。這個想法是將關鍵數據存儲在 AWS 秘密中，並參考YAML配置中的秘密，以便在 Apache Tomcat 啟動時即時拾取秘密內容。

Aurora 的秘密

Aurora 資料庫組態 (針對JICS、Blusam、客戶資料庫等) 將使用內建的資料庫密碼，這會自動從對應的資料庫填入所有相關欄位。

> **Note**
>
> 密dbname鑰是可選的，具體取決於您的數據庫配置，它將進入秘密與否。您可以手動將其添加到該文件中，也可以通過向YAML文件提供名稱。

其他秘密

其他密碼適用於具有單一密碼的資源（特別是受密碼保護的 redis 快取）。在這種情況下，必須使用其他類型的密碼。

YAML對秘密的引用

`application-main.yml`可以引ARN用各種資源的秘密：

JICS資料庫

JICS資料庫認證 `spring.aws.jics.db.secret`

```
spring:
   aws:
     jics:
       db:
         dbname: jics
         secret: arn:aws:secretsmanager:XXXX
```

支援的JICS資料庫秘密金鑰：

私密金鑰	秘密金鑰說明
託管	主機名
port	該端口
dbname	數據庫的名稱

私密金鑰	秘密金鑰說明
使用者名稱	使用者名稱
密碼	密碼
engine	數據庫引擎：郵政，甲骨文，Db2，Microsoft 服務器 SQL
currentSchema	要使用的特定結構描述 (僅支援 Db2)
sslConnection	是否使用SSL連接（僅支持 Db2）
sslTrustStore位置	用戶端上信任庫的位置 (僅限 Db2 支援)
sslTrustStore密碼	用戶端上信任庫的密碼 (僅限 Db2 支援)

> **Note**
>
> 資料庫的名稱是在密碼或 yaml 參考`spring.aws.jics.db.dbname`中提供。

布魯桑數據庫

使用 Blusam 資料庫認證 `spring.aws.client.bluesam.db.secret`

```
spring:
   aws:
     client:
       bluesam:
         db:
           dbname: bluesam
           secret: arn:aws:secretsmanager:XXXX
```

支援的 Blusam 資料庫秘密金鑰：

私密金鑰	秘密金鑰說明
託管	主機名

私密金鑰	秘密金鑰說明
port	該端口
dbname	數據庫的名稱
使用者名稱	使用者名稱
密碼	密碼
engine	數據庫引擎：郵政，甲骨文，Db2，Microsoft 服務器 SQL
currentSchema	要使用的特定結構描述 (僅支援 Db2)
sslConnection	是否使用SSL連接（僅支持 Db2）
sslTrustStore位置	用戶端上信任庫的位置 (僅限 Db2 支援)
sslTrustStore密碼	用戶端上信任庫的密碼 (僅限 Db2 支援)

> **Note**
>
> 資料庫的名稱是在密碼或 yaml 參考`spring.aws.client.bluesam.db.dbname`中提供。

用戶端資料

用戶端`application-profile.yml`可以參考用戶端資料庫的密碼ARN。這需要一個額外的屬性來列出數據源名稱。`spring.aws.client.datasources.names`對於每個資料來源名稱，請在下列屬性ARN中`ds_name`指定密碼：`spring.aws.client.datasources.ds_name.secret`範例：

```
spring:
   aws:
     client:
       datasources:
         names: primary,host
         primary:
           secret: arn:aws:secretsmanager:XXXX
         host:
```

```
            dbname: hostdb
            secret: arn:aws:secretsmanager:XXXX
```

名稱:主, 主機:

範例包含兩個名為 [主要] 和 [host] 的用戶端資料來源，每個資料來源都有其資料庫和認證。

數據庫名稱：主機數據庫：

在這個例子中，「主機」數據庫的名稱不在秘密中，而是在這裡提供，而對於「主」數據庫來說，它是秘密。

支援的用戶端資料庫密鑰：

私密金鑰	秘密金鑰說明
託管	主機名
port	該端口
dbname	數據庫的名稱
使用者名稱	使用者名稱
密碼	密碼
engine	數據庫引擎：郵政，甲骨文，Db2，Microsoft 服務器 SQL
currentSchema	要使用的特定結構描述 (僅支援 Db2)
sslConnection	是否使用SSL連接（僅支持 Db2）
sslTrustStore位置	用戶端上信任庫的位置 (僅限 Db2 支援)
sslTrustStore密碼	用戶端上信任庫的密碼 (僅限 Db2 支援)

PGM工具資料庫

`application-utility-pgm.yml`可以引ARN用各種資源的秘密。

- spring.aws.client.datasources.primary
  - secret

    應用程ARN式資料庫的密碼。

類型：字串

- type

要使用的連線集區實作的完整名稱。

類型：字串

預設：com.zaxxer.hikari.HikariDataSource

- spring.aws.client.utility.pgm.datasources
  - names

資料來源名稱清單。

類型：字串

- dsname
  - dbname

主機的名稱。

類型：字串

- secret

主機資料庫ARN的秘密。

類型：字串

- type

要使用的連線集區實作的完整名稱。

類型：字串

預設：com.zaxxer.hikari.HikariDataSource

對於多數據源秘密：

```
spring:
   aws:
     client:
       primary:
         secret: arn:aws:secretsmanager:XXXX
         type: dataSourceType
       utility:
         pgm:
           datasources:
             names: dsname1,dsname2,dsname3
               dsname1:
                 dbname: dbname1
                 secret: arn:aws:secretsmanager:XXXX
                 type: dataSourceType
               dsname2:
                 dbname: dbname2
                 secret: arn:aws:secretsmanager:XXXX
                 type: dataSourceType
               dsname3:
                 dbname: dbname3
                 secret: arn:aws:secretsmanager:XXXX
                 type: dataSourceType
```

沒有 XA 支援的私密金鑰

- 引擎（後期/Oracle 的/數據庫 2 /mssql）
- port
- dbname
- currentSchema
- 使用者名稱
- 密碼
- url
- sslConnection
- sslTrustStore位置

- sslTrustStore密碼

postgres只有sslMode密鑰值（disable/allow/prefer/require/verify-ca/verify-full）和spring.aws.rds.ssl.cert-pathYAML屬性使其可以與連接SSL。

XA 支援的密鑰

如果用戶端資料庫使用 XA，則會透過秘密值支援子 xa 屬性。

- 託管
- port
- dbname
- currentSchema
- 使用者名稱
- 密碼
- url
- sslConnection （真/假）
- sslTrustStore位置
- sslTrustStore密碼

但是，對於其他 xa-properties（例如maxPoolSize或driverType），仍然spring.jta.atomikos.datasource.XXXX.unique-resource-name必須提供常規YAML密鑰。

密碼值會覆寫YAML屬性。

默認超級管理員BAC和 JAC

您也可以設定應用/main.yml，藉由指定來擷取秘密密碼中預設超級管理員使用者的使用者名稱和AWS密碼。ARN下列範例會示範如何在YAML檔案中宣告此機密。

```
spring:
   aws:
     client:
       defaultSuperAdmin:
         secret: arn:aws:secretsmanager:XXXX
```

支援的預設超級管理員資料庫密鑰：

私密金鑰	秘密金鑰說明
使用者名稱	使用者名稱。
密碼	密碼。

OAuth2

您也可以設定 '應用程式 main.yml`，藉由指定提供者和來 AWS Secrets Manager 擷取用OAuth2戶端密碼。ARN提供者屬性的預設值為 Amazon Cognito。以下是OAuth2提供者 Keycloak 的範例組態：

```
spring:
   aws:
     client:
       provider: keycloak
       keycloak:
         secret: arn:aws:secretsmanager:XXXX
```

在此範例中，會從 Secrets Manager 中指定ARN的擷取OAuth2提供者 Keycloak 的用戶端密碼。AWS此組態透過動態解析提供者名稱和對應的密碼來支援多個提供者ARN。

支援的OAuth2密鑰：

私密金鑰	秘密金鑰說明
用戶端機密	授權伺服器在應用程式註冊過程中產生的密碼。

秘密經理 Redis 的緩存

該`application-main.yml`文件可以引ARN用 Redis 緩存的秘密。支持的一個是：

- 與雷迪斯憑證 `spring.aws.client.gapwalk.redis.secret`
- 藍薩姆雷迪斯憑據 `spring.aws.client.bluesam.redis.secret`
- 藍調鎖定雷迪斯憑據 `spring.aws.client.bluesam.locks.redis.secret`
- 數據集目錄 Redis 的憑據 `spring.aws.client.dataset.catalog.redis.secret`
- JICS雷迪斯的憑據 `spring.aws.client.jics.redis.secret`

- 工作階段 Redis 憑證 `spring.aws.client.jics.redis.secret`
- 會話跟踪器 Redis 的憑據 `spring.aws.client.session.tracker.redis.secret`
- JICSTS 隊列雷迪斯憑據 `spring.aws.client.jics.queues.ts.redis.secret`
- JCL檢查點雷迪斯憑證 `spring.aws.client.jcl.checkpoint.redis.secret`
- 間隙行走文件鎖定 Redis 憑據 `spring.aws.client.gapwalk.files.locks.redis.secret`
- 藍 4iv 鎖定雷迪斯憑據 `spring.aws.client.blu4iv.locks.redis.secret`

下列範例顯示如何在YAML檔案中宣告這些密碼。

```
spring:
   aws:
     client:
       gapwalk:
         redis:
           secret: arn:aws:secretsmanager:XXXX
       bluesam:
         locks:
           redis:
             secret: arn:aws:secretsmanager:XXXX
         redis:
           secret: arn:aws:secretsmanager:XXXX
       dataset:
         catalog:
           redis:
             secret: arn:aws:secretsmanager:XXXX
       jics:
         redis:
           secret: arn:aws:secretsmanager:XXXX
       session:
         tracker:
           redis:
             secret: arn:aws:secretsmanager:XXXX
       jics:
         queues:
           ts:
             redis:
               secret: arn:aws:secretsmanager:XXXX
       jcl:
         checkpoint:
           redis:
             secret: arn:aws:secretsmanager:XXXX
```

```
      gapwalk:
        files:
          locks:
            redis:
              secret: arn:aws:secretsmanager:XXXX
      blu4iv:
        locks:
          redis:
            secret: arn:aws:secretsmanager:XXXX
```

支援的 Redis 秘密金鑰：

私密金鑰	秘密金鑰說明
hostName	Redis 伺服器的主機名稱。
port	Redis 的伺服器連接埠。
使用者名稱	使用者名稱。
密碼	密碼。

密碼設置的SSL密碼管理器

該application-main.yml文件可以引ARN用密SSL碼設置的密碼。支持以下內容。

- 與 Gapwalk SSL 認證 spring.aws.client.ssl.secret

下列範例顯示如何在YAML檔案中宣告這些密碼。

```
spring:
   aws:
     client:
       ssl:
         secret: arn:aws:secretsmanager:XXXX
```

私密金鑰	秘密金鑰說明
trustStorePassword	信任庫密碼。

私密金鑰	秘密金鑰說明
keyStorePassword	金鑰儲存庫密碼。

IBMMQ 密碼設定的密碼管理員

該application-main.yml文件可以引ARN用 IBM MQ 密碼設置的密碼。支持以下內容。

- IBMMQ 連線會定義為清單，認證也會定義為：

  mq.queues.jmsMQQueueManagers[N].secret:

  N 從 0 開始對於第一個連接。

下列範例顯示如何在YAML檔案中宣告這些密碼。

```
mq.queues.jmsMQQueueManagers[0].secret: Secret-0-ARN
mq.queues.jmsMQQueueManagers[1].secret: Secret-1-ARN
```

如需密碼的相關資訊ARNs，請參閱「秘 Secrets Manager 密碼中有什麼內容？

私密金鑰	秘密金鑰說明
密碼	IBMMQ 佇列管理員密碼。

其他文件（常規，SQL 等）

客戶專案使用的其他檔案使用與彈簧規劃相似的優先順序規則。範例：

- Groovy 腳本是.groovy文件scripts夾或子文件夾中的文件。
- SQL腳本是sql資料夾或子資料夾中的.sql檔案。
- 守護進程腳本是.groovy文件daemons夾或子文件夾中的文件。
- 查詢數據庫映射文件是文件夾子queries-database.mapping文件sql夾中名為文件的文件。
- 碧玉模板是.jrxml文件templates夾或子文件夾中的文件。
- 資料集目錄是資料catalog夾中的.json檔案。
- Lnk 文件是文件夾中的.jsonlnk文件。

所有這些位置都可以通過系統屬性或客戶端YAML屬性來覆蓋。

- 對於常規腳本：configuration.scripts
- 對於SQL腳本：configuration.sql
- 對於守護進程腳本：configuration.daemons
- 對於查詢資料庫對應檔案：configuration.databaseMapping
- 對於碧玉模板：configuration.templates
- 對於資料集目錄：configuration.catalog
- 對於 Lnk 檔案：configuration.lnk

如果找不到該屬性，則將從上面提到的默認位置獲取文件。查找將首先以 tomcat 工作目錄作為根目錄完成，最後在應用程序 war 文件中完成。

其他 Web 應用程式

AWS 藍光時代運行時在其webapps-extra文件夾中包含其他 Web 應用程序。這些應用程序默認情況下，tomcat 服務器不提供服務。

選擇使用這些 Web 應用程序是依賴於現代化項目，並通過將所需的 war 文件從文件夾移動到文件webapps-extra夾來完成。webapps之後，Tomcat 服務器將在下次啟動時提供戰爭。

一些特定於項目的附加配置也可以添加到每個額外的戰爭的YAML配置文件中，如application-main.yml文件中所做的，並在上面說明。額外的戰爭是：

- gapwalk-utility-pgm.war：包含對實用程ZOS序的支持，並將其用application-utility-pgm.yaml作其配置。
- gapwalk-cl-command.war：包含 AS/400 公用程式的支援，並用application-cl-command.yaml作其組態。
- gapwalk-hierarchical-support.war：包含IMS/MFS事務支持，並用application-jhdb.yaml作其配置

啟用 AWS 藍光時代運行時的屬性

在 Spring Boot 應用程序application-main.yml是配置文件中，我們定義不同類型的屬性，如監聽端口，數據庫連接性等等。您可以使用此頁面來瞭解 AWS Blu Age 執行階段的可用特性，以及如何啟用它們。

主題

- YML符號
- 快速入門/使用案例
- 主要應用程式的可用屬性
- 可選 Web 應用程式的可用屬性

YML符號

在下列文件中，如下所示`parent.child1.child2=true`的屬性會以YAML格式寫入。

```
parent:
 child1:
   child2: true
```

快速入門/使用案例

下列使用案例顯示適用索引鍵和值的範例。

- 默認應用程序主要 .yml

```
----
#### DEFAULT APPLICATION-MAIN.YML FILE      #####
#### SHOWING USEFUL CONFIGURATION ELEMENTS #####
#### SHOULD BE OVERRIDDEN AND EXTERNALIZED   #####

##################################
##### Logging configuration #####
##################################
logging:
  config: classpath:logback-main.xml
  level.org.springframework.beans.factory.support.DefaultListableBeanFactory : WARN

##################################
##### Spring configuration #####
##################################
spring:
  quartz:
    auto-startup: false
    scheduler-name: Default
    properties:
```

```
      org.quartz.threadPool.threadCount: 1
  jta:
    enabled: false
    atomikos.properties.maxTimeout : 600000
    atomikos.properties.default-jta-timeout : 100000
  jpa:
# DISABLE OpenEntityManagerInViewInterceptor
    open-in-view: false
    # Fix Postgres JPA Error:
    # Method org.postgresql.jdbc.PgConnection.createClob() is not yet implemented.
    properties.hibernate.temp.use_jdbc_metadata_defaults : false
#####################################
##### Jics tables configuration #####
#####################################

    # The dialect should match the jics datasource choice
    database-platform : org.hibernate.dialect.PostgreSQLDialect #
 org.hibernate.dialect.PostgreSQLDialect, org.hibernate.dialect.SQLServerDialect

    # those properties can be used to create and initialize jics tables
 automatically.
#    properties:
#      hibernate:
#      globally_quoted_identifiers: true
#        hbm2ddl:
#           import_files_sql_extractor :
 org.hibernate.tool.hbm2ddl.MultipleLinesSqlCommandExtractor
#           import_files : file:./setup/initJics.sql
#           auto : create

#########################
###### Level 2 cache #####
#########################
#        cache:
#          use_second_level_cache: true
#          use_query_cache: true
#          region:
#            factory_class: org.hibernate.cache.ehcache.EhCacheRegionFactory
#      javax:
#        persistence:
#          sharedCache:
#            mode: ENABLE_SELECTIVE
#########################
###### Redis settings #####
```

```
##########################
  session:
    store-type: none #redis

########################################
##### JICS datasource configuration #####
########################################
datasource:
  jicsDs:
    driver-class-name : org.postgresql.Driver # org.postgresql.Driver,
 com.microsoft.sqlserver.jdbc.SQLServerDriver
    url: jdbc:postgresql://localhost/jics # jdbc:postgresql://localhost:5433/jics,
 jdbc:sqlserver://localhost\SQLEXPRESS:1434;databasename=jics;
    username: jics
    password: jics
    type : org.postgresql.ds.PGSimpleDataSource #
 org.postgresql.ds.PGSimpleDataSource,
 com.microsoft.sqlserver.jdbc.SQLServerDataSource

#####################################################
##### Embedded Bluesam datasource configuration #####
#####################################################
  bluesamDs :
    driver-class-name : org.postgresql.Driver # org.postgresql.Driver,
 com.microsoft.sqlserver.jdbc.SQLServerDriver
    url : jdbc:postgresql://localhost/bluesam # jdbc:postgresql://localhost:5433/
jics, jdbc:sqlserver://localhost\SQLEXPRESS:1434;databasename=jics;
    username : bluesam
    password : bluesam
    type : org.postgresql.ds.PGSimpleDataSource #
 org.postgresql.ds.PGSimpleDataSource,
 com.microsoft.sqlserver.jdbc.SQLServerDataSource

########################################
##### Embedded Bluesam configuration #####
########################################
bluesam :
  remote : false
  cache : ehcache
  persistence : pgsql #pgsql, mssql, xodus...
  ehcache:
    resource-pool:
      size: 4GB
  write-behind:
```

```
    enabled: true
  pgsql :
    dataSource : bluesamDs

#########################
##### Jics settings #####
#########################
rabbitmq.host: localhost
jics:
  cache: false #redis
  resource-definitions.store-type: jpa # default value: jpa, other possible value:
 redis
  redis.hostname: 127.0.0.1 # Redis server host.
  redis.password: redis # Login password of the redis server.
  redis.port: 6379 # Redis server port.
  redis.username: # Redis username
  redis.mode: standalone # Redis mode. Possible values: standalone, cluster
jics.disableSyncpoint : false
#jics.initList:
#jics.parameters.datform: DDMMYY
#jics.parameters.applid: VELOCITY
#jics.parameters.sysid: CICS
#jics.parameters.eibtrmid: TERM
#jics.parameters.userid: MYUSERID
#jics.parameters.username: MYUSERNAME
#jics.parameters.opid: XXX
#jics.parameters.cwa.length: 0
#jics.parameters.netname: MYNETNAME
#jics.parameters.jobname: MJOBNAME
#jics.parameters.sysname: SYSNAME

###############################################
##### Jics RunUnitLauncher pool settings #####
###############################################
#jics.runUnitLauncherPool.enable: false
#jics.runUnitLauncherPool.size: 20
#jics.runUnitLauncherPool.validationInterval: 1000

#########################
##### Jhdb settings #####
#########################
#jhdb.lterm: LTERMVAL
#jhdb.identificationCardData: SomeIDData
```

```
###################################
##### DateHelper configuration ####
###################################
#forcedDate: "2013-08-26T12:59:58+01:57"

#############################
##### Sort configuration ####
#############################
#externalSort.threshold: 256MB

###################################
##### Server timeout (10 min)  ####
###################################
spring.mvc.async.request-timeout: 600000

#############################
##### DATABASE STATISTICS  ####
#############################
databaseStatistics : false

######################
##### CALLS GRAPH ####
######################
callGraph : false

###################################
#####     SSL configuration    #####
###################################
gapwalk.ssl.enabled : true
gapwalk.ssl.trustStore : "./config/clientkey.jks"
gapwalk.ssl.trustStorePassword : mysslcertifpassword

#################################
##### MQ settings #####
#################################
mq.queues: jmsmq
mq.queues.jmsMQQueueManagers[0].jmsMQQueueManager: QM1
mq.queues.jmsMQQueueManagers[0].jmsMQAppName: Gapwalk
mq.queues.jmsMQQueueManagers[0].jmsMQChannel: DEV.APP.SVRCONN
mq.queues.jmsMQQueueManagers[0].jmsMQHost: localhost
mq.queues.jmsMQQueueManagers[0].jmsMQPort: 1415
mq.queues.jmsMQQueueManagers[0].jmsMQUserid: app
mq.queues.jmsMQQueueManagers[0].jmsMQSSLCipher: "*TLS12ORHIGHER"
mq.queues.jmsMQQueueManagers[1].jmsMQQueueManager: QM2
```

```
mq.queues.jmsMQQueueManagers[1].jmsMQAppName: Gapwalk
mq.queues.jmsMQQueueManagers[1].jmsMQChannel: DEV.APP.SVRCONN
mq.queues.jmsMQQueueManagers[1].jmsMQHost: localhost
mq.queues.jmsMQQueueManagers[1].jmsMQPort: 1415
mq.queues.jmsMQQueueManagers[1].jmsMQUserid: app

################################
##### SQL SHIFT CODE POINT ####
################################
# Code point 384 match unicode character \u0180
sqlCodePointShift : 384

################################
##### LOCK TIMEOUT RECORD  ####
################################
# Blu4IV record lock timeout
lockTimeout : 100

################################
##### REPORTS OUTPUT PATH ####
################################
reportOutputPath: reports

################################
##### TASK EXECUTOR       ####
################################
taskExecutor:
  corePoolSize: 5
  maxPoolSize: 10
  queueCapacity: 50
  allowCoreThreadTimeOut: false

################################
##### PROGRAM NOT FOUND   ####
################################
stopExecutionWhenProgNotFound: false

#########################################################
#####  DISP DEFAULT VALUE (to be removed one day) ####
#########################################################
defaultKeepExistingFiles: true

#####################################
#####  JOBQUEUE CONFIGURATION    ####
```

```
####################################
jobqueue:
  api.enabled: false
  impl: none # possible values: quartz, none
  schedulers:  # list of schedulers
    -
      name: queue1
      threadCount: 5
    -
      name: queue2
      threadCount: 5

################################################################################
##### QUERY BUILDING                                                       ##
# useConcatCondition : false by default
# if true, in the query, the where condition is build with key concatenation  ##
################################################################################
# query.useConcatCondition: true
----
```

- 搭配LISTCAT指令使用可變長度檔案

```


[**/*.*]
encoding=IBM930
reencoding=false

[global]
listcat.variablelengthpreprocessor.enabled=true
listcat.variablelengthpreprocessor.type=rdw
# use "rdw" if your .listcat file contains a set of records (RDW)
# use "bdw" if your .listcat file contains a set of blocks (bdw)
```

- 在/實用UNLOAD程序中提供空字節指示LOAD器值

```


# Unload properties
# For date/time: if use database configuration is enabled, formats are ignored
# For nbi; use hexadecimal syntax to specify the byte value
# - When the value is null in database : the value dumped to the file is filled by
 low value characters and the NBI is
```

```
# equal to the byte 6F (the ? character)
# - When the value is not null in database and the column is nullable: the NBI is
 equal to the byte 00 (low value) and NOT
# equal to the byte 40 (space)
unload:
  sqlCodePointShift: 0
  nbi:
    whenNull: "6F"
    whenNotNull: "00"
  useDatabaseConfiguration: false
  format:
    date: MM/dd/yyyy
    time: HH.mm.ss
    timestamp: yyyy-MM-dd-HH.mm.ss.SSSSSS
```

主要應用程式的可用屬性

此表格提供索引鍵/值參數的詳盡檢視。

金鑰	Type	預設值	描述
`logging.config`	路徑	類路徑:記錄檔主 .xml	登入組態檔案參照的標準金鑰。其他標準記錄鍵也可用。
`spring.jta.enabled`	boolean	false	標準鍵。如果數據源支持模式不是靜態的 xa，則必須禁用 spring JTA 事務自動配置。
`datasource.jicsDs + -driver-class-name + -url + -username + -password + -type`	帶有子鍵的標準彈簧數據源		包含 Jics 資料庫的連線資訊。或者，強烈鼓勵使用 AWS 秘密，如中the section called “JICS資料庫”所述。

金鑰	Type	預設值	描述
datasource.bluesamDs + -driver-class-name + -url + -username + -password + -type	帶有子鍵的標準彈簧數據源		包含 Blusam 資料庫的連線資訊。或者，強烈鼓勵使用 AWS 秘密，如中the section called “布魯桑數據庫”所述。
bluesam.disabled	boolean	false	是否完全禁用布魯薩姆。
bluesam.cache	string		如果未設置，則不會使用 Blusam 緩存。可能的值（緩存實現）是緩存和 redis。
forcedDate	string		強制日期提供的日期，如果有一個。
frozenDate	布林值	true	指定是否凍結日期。僅在同時設forcedDate 定時套用。
externalSort.threshold	數據化（例如：12 MB）		排序閾值：何時切換到外部（合併）排序。
jics.parameters.datform	string	MMDDYY	日期表單。

金鑰	Type	預設值	描述
jics.initList	string		初始化JICS清單，以逗號分隔。如果存在，它定義列表的逗號分隔的名稱，以在 Apache Tomcat 啟動列表中CICS激活。範例值：UUU,DFHIVPL,PEZ1 。這將重疊顯示到這些列表中包含的組及其基礎資源定義，然後運行時可以看到這些組。預設為空白。
jics.parameters.applid	string	VELOCITY	套用於識別應用程式 JICS (至少 4 個字元，沒有最大長度)。
jics.parameters.sysid	string	CICS	系統識別 (SYSID)。
jics.parameters.eibtrmid	string	TERM	終端標識符 (最多 4 個字元，最少 1 個字元) 。
jics.parameters.userid	string		使用者 ID (最多 8 個字元，不得最少)。如果未提供任何值 (預設為空白)，則會使用 HTTP工作階段 ID 做為使用者 ID。
jics.parameters.username	string	MYUSERNAME	使用者名稱 (最多 10 個字元，最少 1 個字元)。

金鑰	Type	預設值	描述
jics.parameters.netname	string	MYNETNAME	網路名稱 (最多 8 個字元，最少 1 個字元)。
jics.parameters.opid	string	XXX	3 個字元的運算子識別。
jics.parameters.jobname	string	MJOBNAME	工作名稱。
jics.parameters.sysname	string	SYSNAME	AS400 系統名稱 (系統名稱)。
jics.parameters.cwa.length	number	0	公共工作區域的長度（CWA）。
jics.parameters.charset	string	CP037	JICS全域使用的字元集。
jics.parameters.tsqimpl	string	藍色攝影機	JICS臨時存儲隊列（TSQ）實現（允許的值是bluesam/memory/redis）
jics.queues.ts.redis.hostname	string	127.0.0.1	jics 快取 redis 伺服器的主機名稱。
jics.queues.ts.redis.port	number	6379	jics 快取記憶體伺服器的連接埠。
jics.queues.ts.redis.password	string	redis	jics 快取記憶體的伺服器密碼。

金鑰	Type	預設值	描述
jics.queues.ts.redis.username	string		jics 快取 Redis 伺服器的使用者名稱。預設值為空白 (無使用者名稱)。
jics.queues.ts.redis.mode	string	獨立	jics 緩存模式。可能的值為standalone 或cluster。預設值為standalone 。
lockTimeout	number	500	鎖定逾時，以毫秒為單位。
sqlCodePointShift	number		選用。SQL 代碼點移位。將舊RDBMS數據遷移到現代數據時，我們可能會遇到的控制字符的代RDBMS碼點。例如，您可以指定384要符合 Unicode 字元\u0180。
sqlIntegerOverflowAllowed	boolean	false	指定是否允許SQL整數溢位，表示是否允許在 host 變數中放置較大的值。

金鑰	Type	預設值	描述
database.cursor.overflow.allowed	布林值	true	指定是否允許游標溢位。設定true為在游標上執行下一個呼叫，無論其位置為何。設定false為可在對游標執行下一次呼叫之前檢查游標是否在最後一個位置。只有在游標為 SCROLLABLE (SENSITIVE或INSENSITIVE) 時才啟用。
reportOutputPath	string	/reports	報告輸出路徑。
spring.session.store-type	string	無	高可用性環境的工作階段快取。可能的值為none或redis。預設值為none。
stopExecutionWhenProgNotFound	布林值	true	指定如果找不到程式，是否停止執行。如果設定為true，則在找不到程式時中斷執行。
forceHR	boolean	false	指定是否在主控台或檔案輸出上使用「人類可讀SYSPRINT」。
rollbackOnRTE	boolean	false	指定是否針對執行階段例外狀況復原隱含執行單元交易。

金鑰	Type	預設值	描述
sctThreadLimit	long	5	觸發指令碼的執行緒限制。
dataSimplifier.onInvalidNumericData	string	拒絕	解碼無效數字數據時如何做出反應。允許的值為 rejecttolerates paces //tolerates paceslowv alues /toleratem ost 。預設值為reject。
filesDirectory	string		批次輸入/輸出檔案的目錄。
ims.messages.extendedSize	boolean	false	指定是否設定IMS郵件的延伸大小。
defaultKeepExistingFiles	boolean	false	指定是否設定資料集預設先前值。
jics.db.ddlScriptLocation	string		Jics DDL 指令碼位置。可讓您使用 .sql 命令檔起始 Jics 資料庫結構描述。預設為空白。例如，./jics/sql/jics.sql 。

金鑰	Type	預設值	描述
jics.db.schemaTestQueryLocation	string		應包含唯一查詢的 sql 檔案位置，該查詢會傳回 jics 結構描述 (如果有的話) 中的物件數目。
jics.db.dataScriptLocation	string		initJics.sql 指令碼的位置，由分析器從大型主機剖析CSD匯出時準備。
jics.db.dataTestQueryLocation	string		包含單一 sql 查詢的 sql 指令碼位置，該查詢預期會傳回物件計數 (例如：計算 jics 程式資料表中的記錄數目)。如果計數等於 0，數據庫將使用jics.db.dataScriptLocation 腳本加載，否則數據庫負載將被跳過。
jics.data.dataJsonInitLocation	string		
jics.xa.agent.timeout	number		
query.useConcatCondition	boolean	false	指定鍵條件是否由密鑰串聯或不構建。
system.qdecfmt	string		

金鑰	Type	預設值	描述
disposition.checkexistence	boolean	false	指定是否要針對使用DISPSHR或的資料集核發檔案是否存在的檢查OLD。
useControlMVariable	boolean	false	指定是否要使用Control-M 規格進行變數取代。
card.encoding	string	CP1145	卡編碼：與useControlMVariable .
mapTransfo.prefixes	string	&,@,%%	轉換 ControlM 變量時要使用的前綴列表。每一個用逗號分隔。
checkinputfilesize	boolean	false	指定檔案大小是否為記錄大小的倍數，是否要釋放檢查。
stepFailWhenAbend	布林值	true	指定是否在步驟失敗或完成執行時引發異常結束。
bluesam.fileLoading.commitInterval	number	100000	藍光提交間隔。
uppercaseUserInput	布林值	true	指定使用者輸入是否必須為大寫。

金鑰	Type	預設值	描述
jhdb.lterm	string		允許您在IMS模擬的情況下強制使用通用邏輯終端機 ID。如果沒有設置 sessionId 則使用。
jhdb.identificationCardData	string		用於將某些「操作員識別卡數據」硬編碼到CARD參數指定的MID字段中。默認為空白，沒有輸入限制。
encoding	string	ASCII	項目中使用的編碼（不在 groovy 文件中）。期望有效的編碼CP1047IBM930、ASCII、、
cl.configuration.context.encoding	string	CP297	CL 檔案的編碼。期望有效的編碼CP1047IBM930、ASCII、、預設值為 CP297
cl.zonedMode	string	EBCDIC_STRICT	編碼或解碼控制語言 (CL) 指令的模式。允許的值是EBCDIC_STRICT /EBCDIC_MODIFIED /AS400。

金鑰	Type	預設值	描述
ims.programs	string		要使用的IMS程式清單。使用分號 () 分隔每個參數，並以逗號 (;) 分隔每個交易。,例如：PCP008,PCT008;PCP054,PCT054;PCP066,PCT066;PCP068,PCT068;
jhdb.configuration.context.encoding	string	CP297	JHDB(Java 階層式資料庫) 編碼。需要有效的編碼字串CP1047IBM930、ASCII、、
jhdb.metadata.extrapath	string	文件：。 /設置/	為 psbs 和 dbds 資料夾指定額外、執行階段特定的根資料夾的組態參數。

金鑰	Type	預設值	描述
jhdb.checkpointPersistence	string	無	檢查點持久性模式。允許的值是none/add/end。用add於在建立新檢查點並將其新增至登錄時保留檢查點。在服務器關閉時使用end太持久檢查點。任何其他值都會停用持續性。請注意，每次將新的檢查點添加到註冊表時，所有現有的檢查點都將被序列化，並且該文件將被刪除。它不是附加到文件中的現有數據。因此，根據檢查站的數量，它可能會對性能產生一些影響。

金鑰	Type	預設值	描述
jhdb.checkpointPath	string	文件：。 /設置/	如果不jhdb.checkpointPersistence 是，none則此參數允許您設置檢查點持續性路徑 (checkpoint.dat 文件存儲位置），則註冊表中包含的所有檢查點數據將被序列化並備份到提供的文件夾中的文件（checkpoint.dat）中。請注意，此備份只關注檢查點資料 (scriptIdstepId、、資料庫位置和檢查點區域)。
jhdb.navigation.cachenexts	number	5000	的階層式導覽中使用的快取持續時間 (以毫秒為單位) RDBMS。
jhdb.use-db-prefix	布林值	true	指定是否在的階層式導覽中啟用資料庫前置詞RDBMS。
jhdb.query.limitJoinUsage	布林值	true	指定是否在RDBMS圖形上使用限制聯結用法參數。
taskExecutor.corePoolSize	number	5	當終端中的事務通過 groovy 腳本啟動時，將創建一個新的線程。使用此參數可設定核心集區大小。

金鑰	Type	預設值	描述
taskExecutor.maxPoolSize	number	10	當終端中的事務通過 groovy 腳本啟動時，將創建一個新的線程。使用此參數可設定集區大小上限 (parallel 執行緒的最大數目)。
taskExecutor.queueCapacity	number	50	當終端中的事務通過 groovy 腳本啟動時，將創建一個新的線程。使用此參數可設定佇列大小。(= 達到暫緩交易的最大數目) taskExecutor.maxPoolSize
taskExecutor.allowCoreThreadTimeOut	boolean	false	指定是否允許核心執行緒逾時JCIS。即使與非零佇列結合使用，這也可以實現動態增長和縮小 (因為佇列已滿時，最大池大小才會增加)。
jics.runUnitLauncherPool.enable	boolean	false	指定是否啟動中的執行單元啟動器集區JICS。
jics.runUnitLauncherPool.size	number	20	執行單位啟動器集區大小JICS。

金鑰	Type	預設值	描述
jics.runUnitLauncherPool.validationInterval	number	1000	調整集區大小的工作每次執行之間的間隔。
jics.runUnitLauncherPool.parallelism	number	2	執行調整工作時，用來在佇列中產生遺失的執行個體的執行緒數目。
context.preconstruct.enable	boolean	false	指定是否啟用程式前後關聯的預建構。
context.preconstruct.frequencyInMillis	number	100	調整集區大小的工作每次執行之間的間隔。
context.preconstruct.parallelism	number	5	執行調整工作時，用來在佇列中產生遺失的執行個體的執行緒數目。
context.preconstruct.minInstances	number	2	第一次需要前後關聯時將建立的執行個體數目。
spring.aws.application.credentials	string	null	從中的認證設定檔檔案載入身份證 AWS 明 JICS。

金鑰	Type	預設值	描述
jics.queues.sqs.region	string	eu-west-1	Amazon 簡單佇列服務的 AWS 區域，用於 JICS.
mq.queues.sqs.region	string	eu-west-3	AWSSQSMQ 服務的 AWS 區域。
quartz.scheduler.stand-by-if-error	boolean	false	指定工作排程器處於待命模式時是否觸發工作執行。如果為 true，則不會觸發「啟用時」工作執行。
databaseStatistics	boolean	false	指定是否允許SQL建置工具收集和顯示統計資訊。
dbDateFormat	string	yyyy-MM-dd	數據庫目標日期格式。
dbTimeFormat	string	高:毫米:SS	數據庫目標時間格式。
dbTimestampFormat	string	年-月-日高:毫米:SS.SSSSSS	db 目標時間戳記格式。
dateTimeFormat	string	ISO	dateTimeFormat 說明如何將資料庫日期時間戳記類型溢滿到資料簡化器實體中。允許的值為 ISOEUR//EUR/USA/LOCAL
localDateFormat	string		本機日期格式清單。使用分隔每種格式。\

金鑰	Type	預設值	描述
localTimeFormat	string		本地時間格式列表。分隔每種格式 \
localTimestampFormat	string		本機時間戳記格式的清單。使用分隔每種格式\。
pgmDateFormat	string	yyyy-MM-dd	日期時間格式。
pgmTimeFormat	string	HH. 毫米	pgm（程序）執行所使用的時間格式。
pgmTimestampFormat	string	年-毫米-日-赫姆。SSSSSS	時間戳記格式。
cacheMetadata	布林值	true	指定是否快取資料庫中繼資料。
forceDisableSQLTrimStringType	boolean	false	指定是否停用所有 sql 字串參數的修剪。
fetchSize	number		游標的 fetchSize 值。通過加載/卸載實用程序使用塊獲取數據時使用。
check-groovy-file	布林值	true	指定是否在註冊之前檢查 groovy 文件的內容。
qtemp.uuid.length	number	9	QTEMP唯一的 ID 長度。
qtemp.dblog	boolean	false	是否啟用QTEMP數據庫日誌記錄。

金鑰	Type	預設值	描述
qtemp.cleanup.threshold.hours	number	0	指定何qtemp.dblog 時啟用。db 分區存留期（以小時為單位）。
sort.function	string		blu4iv 資料庫的排序函數名稱。
invalidDataTolerence	布林值	true	指定是否允許封裝類型的無效資料。
program.timeout	number	-1	指定任何程式/事務執行的逾時時間 (以秒為單位)。在此時間之後，系統將嘗試中斷程序。
gapwalk.line.separator	string	null	指定間隙行道中的線分隔符號類型。允許的值為 WIN (CRLF)/UNIX(LF)/LINUX(LF)。其他值將被忽略，並使用系統行 .sepatator 屬性。
enableActivePgmIdCache	boolean	false	指定是否啟用作用中的程式 ID 本機快取。請小心使用此功能，因為JICS資源可以在程式和使用者之間共用。任何系統管理員都可以在外部變更這些資源，而且放置的本機快取可能會失效。

金鑰	Type	預設值	描述
mq.queues .default. syncpoint	boolean	false	指定未設定 MQPMO _ SYNCPOINT 或 MQPMO _NO_ SYNCPOINT 時 MQ 指PUT令的 預設行為。當設 置為 true 時，它 就像MQPMO_SYN CPOINT 在PUT命 令期間NOT直接提 交消息一樣。當設 置為 false 時，它 的作用MQPMO_NO_ SYNCPOINT 與消息 在PUT命令期間直接 提交。
dataSimpl ifier.byt eRangeBou ndsCheck	boolean	false	當設置為 true 時， 它可以確保沒有使用 不 ByteRange 正確 的值創建。預設值為 false。
file.stdo utIntoLogger	boolean	false	指定是否啟用寫入記 錄器，而不是默認 的系統輸出流在默 認SYSPRINT和SYSPUNCH文 件。
tempFiles Directory	string	null	指定產生之暫存檔案 的資料夾位置名稱。
cleanTemp FilesDire ctoryAtStartup	布林值	true	指定是否在應用程式 啟動時清除暫存檔案 資料夾的內容。

金鑰	Type	預設值	描述
tempFolderPattern	string	null	指定將用於根據下列預先定義和可自訂的資訊動態建立暫存資料夾名稱的模式。 HOST：主機名稱。 JOBID：工作的識別碼。 HASHCODE：工作相關資訊環境的雜湊碼。 TIMESTAMP：獲取時間戳時使用的模式。暫存資料夾的目標名稱是 TMP DIR _ _ {tempFolderPattern }。例如，在以下模式的情況下，該名稱將以作業 ID 開頭，並以「時間戳記」結束 tempFolderPattern：JOBID, HOST = xxxxxHASHCODE, TIMESTAMP = yyyymmddhhmmss。如果屬性tempFolderPattern 未添加到YAML文件中或為空，則臨時文件夾的名稱將是「TMPDIR_ _」+ 此。 hashCode() (DefaultJobContext)。

金鑰	Type	預設值	描述
database.cursor.raise.already.opened.error	boolean	false	指定是否啟用引SQLCODE發錯誤 502 時，已經打開的光標打開。
jics.spool.smtp.hostname	string	null	指定伺SMTP 服器主機。範例：smtp.xxx.com
jics.spool.smtp.port	string	null	指定SMTP伺服器連接埠。範例：25
jics.spool.smtp.password	string	null	指定SMTP伺服器的登入密碼。
jics.spool.smtp.username	string	null	指定SMTP伺服器的使用者名稱。
jics.spool.smtp.debug	boolean	false	指定SMTP伺服器的除錯模式。
gapwalk-application.security	string	disabled	切換全域安全性設定 (XSSCORS、CSRF、、OAUTH 驗證...)。允許的值為 disabled 和 enabled。

金鑰	Type	預設值	描述
gapwalk-application.identity	string	null	全域驗證方法。建議值為oauth。允許的值為 json 和 oauth。如果是，則需要gapwalk-application.security 此選項enabled。
gapwalk-application.security.issuerUri	string	null	身分識別提供者 (IdP) URI 的發行者。如果是，則需要gapwalk-application.identity 此選項oauth。
gapwalk-application.security.allowedOrigins	字符串 []	null	要允許的起源清單。此選項需gapwalk-application.identity 要設定為oauth。
gapwalk-application.security.claimGroupName	string	cognito:groups	包含使用者所屬之所有群組清單的宣告屬性。用cognito:groups 於 Amazon Cognito，或外部 IdP 的任何其他字串。

金鑰	Type	預設值	描述
gapwalk-application.security.userAttributeName	string	username	用來識別使用者要求的宣告屬性名稱。用username於 Amazon Cognito，鍵盤遮罩，preferred_username 或任何其他字符串用於外部 IdP。
gapwalk-application.security.localhostWhitelistingEnabled	布林值	true	指定是否從任何localhost 要求啟用驗證。
gapwalk-application.defaultSuperAdminUserName	string	sadmin	停用gapwalk-application.security 時，指定預設的本機超級使用者名稱。
gapwalk-application.defaultSuperAdminUserPwd	string	sadmin	停用gapwalk-application.security 時，指定預設的本機超級使用者密碼。
gapwalk-application.security.filterURIs	string	disabled	切換過濾URIs 配置。允許的值為 disabled 和 enabled。

金鑰	Type	預設值	描述
gapwalk-application.security.blockedURIs	字符串 []	null	URIs要封鎖的清單。如果是，則需要gapwalk-application.security.filterURIs 此選項enabled。
jics.redis.database	number	0	指定 Redis 伺服器連線處理站的資料庫索引，範圍從 0 到 15。預設值為 0。
jics.redis.maxTotal	number	32	Redis 集區使用中連線的最大數目。
jics.redis.maxIdle	number	32	Redis 池閒置連接的最大數量。
jics.redis.minIdle	number	8	Redis 池閒置連接的最小數量。
gapwalk.ssl.enabled	boolean	false	如果應用程式啟動時尚未設定，則指示將下列gapwalk.ssl.* 屬性設定為目前的JVM系統屬性。
gapwalk.ssl.trustStore	string	null	如果尚未在應用程序啟動時設置該值，javax.net.ssl.trustStore 則將該值設置為系統屬性。

金鑰	Type	預設值	描述
gapwalk.ssl.trustStorePassword	string	null	javax.net.ssl.trustStorePassword 如果尚未在應用程式啟動時設定，請將該值設定為系統屬性。或者，強烈鼓勵使用 AWS 秘密，如中the section called “密碼設置的SSL密碼管理器”所述。
gapwalk.ssl.trustStoreType	string	null	javax.net.ssl.trustStoreType 如果尚未在應用程式啟動時設定，請將該值設定為系統屬性。
gapwalk.ssl.keyStore	string	null	javax.net.ssl.keyStore 如果尚未在應用程式啟動時設定，請將該值設定為系統屬性。

金鑰	Type	預設值	描述
gapwalk.ssl.keyStorePassword	string	null	javax.net.ssl.keyStorePassword 如果尚未在應用程式啟動時設定，請將該值設定為系統屬性。或者，強烈鼓勵使用 AWS 秘密，如中the section called “密碼設置的SSL密碼管理器”所述。
mq.queues	string	sqs	指定使用 Amazon SQS、rabbitmq使用內部部署 Rabbit MQ 或sqs使用內部部署時所支援的佇列代理程式。jms IBMMQ
mq.queues.jmsMQQueueManagers[N]			如果mq.queues 是jms，則可以指定 IBM MQ 連線清單。 mq.queues.jmsMQQueueManagers[0] 對於第一個連接，mq.queues.jmsMQQueueManagers[1] 用於第二個連接，依此類推。

金鑰	Type	預設值	描述
mq.queues.jmsMQQueueManagers[N].jmsMQQueueManager	string	null	IBMMQ佇列管理員名稱。
mq.queues.jmsMQQueueManagers[N].jmsMQAppName	string	null	IBMMQ應用程式名稱。
mq.queues.jmsMQQueueManagers[N].jmsMQChannel	string	null	IBMMQ頻道名稱。
mq.queues.jmsMQQueueManagers[N].jmsMQHost	string	null	IBMMQ主機名稱。
mq.queues.jmsMQQueueManagers[N].jmsMQPort	number	null	端IBMMQ口。
mq.queues.jmsMQQueueManagers[N].jmsMQUserid	string	null	IBMMQ使用者名稱。

金鑰	Type	預設值	描述
mq.queues.jmsMQQueueManagers[N].jmsMQPassword	string	null	IBMMQ使用者密碼。或者，強烈鼓勵使用 AWS 秘密，如中the section called "IBMMQ 密碼設定的密碼管理員"所述。
mq.queues.jmsMQQueueManagers[N].jmsMQMaxPoolSize	number	0	池IBMMQ大小上限。若為 0，則會啟用無限數量的實體連線。
mq.queues.jmsMQQueueManagers[N].jmsMQSSLCipher	string	null	IBMMQSSL密碼套件。一個例子可能是"*TLS12OR HIGHER" 。有關更多詳細信息，請參閱官方文檔TLS CipherSpecs 和 CipherSuites IBM MQ 類。JMS
			如果mq.queues 是rabbitmq，IBMMQ主機名稱。
mq.queues.rabbitMQHost			兔子 MQ 主機名稱。
mq.queues.rabbitMQ VirtualHost			兔子 MQ 虛擬主機名稱。
mq.queues.rabbitMQPort			兔子 MQ 端口。

金鑰	Type	預設值	描述
mq.queues .rabbitMQ Username			兔子 MQ 用戶。
mq.queues .rabbitMQ Password			兔子 MQ 密碼。

可選 Web 應用程式的可用屬性

視您的現代化應用程式而定，您可能需要設定一或多個可選的 Web 應用程式，以代表支援相依性，例如 z/OS、AS/400 或/。IMS MFS下表包含用於設定每個選擇性 Web 應用程式的可用索引鍵/值參數清單。

gapwalk-utility-pgm. 戰爭。

這個選用的 Web 應用程式包含 Z/OS 公用程式的支援。

此表格提供此應用程式之索引鍵/值參數的詳盡檢視。

金鑰	Type	預設值	描述
logging.config	路徑	類路徑:記錄公用程式 .xml	登入組態檔案參照的標準金鑰。其他標準記錄鍵也可用。
spring.jt a.enabled	boolean	false	標準鍵。如果數據源支持模式不是靜態的 xa，則必須禁用彈簧 JTA事務 auto 配置。
spring.da tasource. primary.jndi- name	string	jdbc /主要	主要資料來源的JNDI名稱 (Java 命名和目錄介面) (如果使用的話)。JNDI

金鑰	Type	預設值	描述
primary.datasource -driver-class-name -url -username -password	帶有子鍵的標準彈簧數據源		包含應用程式資料庫的連線資訊 (如果未使用) JNDI。必須具有與現代化應用程式YAML檔案中相同的組態。 或者，強烈鼓勵使用 AWS 秘密，如中the section called “用戶端資料”所述。
encoding	string	ASCII	公用程式中使用的編碼。期望有效的編碼CP1047IBM930、ASCII、、U
sysPunchEncoding	string	ASCII	敘述編碼字元集。期望有效的編碼CP1047IBM930、ASCII、、
zonedMode	string	EBCDIC_STRICT	編碼或解碼分區資料類型的模式。允許的值是EBCDIC_STRICT /EBCDIC_MODIFIED /AS400。
unload.chunkSize	number	0	用於卸載公用程式的區塊大小。
unload.sqlCodePointShift	number	0	卸載實用SQL程序的代碼點移位。運行移位字符過程。當你的目標數據庫DB2是Postgreql 時需要的。
unload.columnFiller	string	空格	卸載實用程序列填充物。

金鑰	Type	預設值	描述
unload.varCharIsNull	boolean	false	在INFTILB程序中使用此參數，如果設置為true則所有不可為空的字段（空格）值返回一個空字符串。
unload.useDatabaseConfiguration	boolean	false	指定是否要在卸載公用程式中使用應用程式 main.yml 的日期或時間組態。
unload.format.date	string	MM/dd/yyyy	如果啟unload.useDatabaseConfiguration 用，則在卸載公用程式中使用的日期格式。
unload.format.time	string	HH. 毫米	如果啟unload.useDatabaseConfiguration 用，卸載公用程式中使用的時間格式。
unload.format.timestamp	string	年-毫米-日-赫姆。SSSSSS	如果啟unload.useDatabaseConfiguration 用，則在卸載公用程式中使用的時間戳記格式。
unload.nbi.whenNull	十六進位	6F	空字節指示器（NBI）值時從數據庫中的值為空添加。

金鑰	Type	預設值	描述
unload.nbi.whenNotNull	十六進位	00	當數據庫中的值不為空時，要添加的空字節指示符（NBI）值。
unload.nbi.writeNullIndicator	boolean	false	指定是否要在卸載輸出檔案中寫出 null 指標。
unload.fetchSize	number	0	可讓您在卸載公用程式中處理游標時調整擷取大小。
treatLargeNumberAsInteger	boolean	false	指定是否將大數字視為Integer。默認情況BigDecimal 下，它們被視為。
load.batchSize	number	0	載入公用程式批次大小。
load.format.localDate	string	日月/年\ 日/月/年\ 年-月-日	要使用的載入公用程式本機日期格式。
load.format.localTime	string	高:毫米:SS\ hh.mm.ss	要使用的載入公用程式本地時間格式。
load.format.dbDate	string	yyyy-MM-dd	要使用的載入公用程式資料庫格式。
load.format.dbTime	string	高:毫米:SS	要使用的載入公用程式資料庫時間。
load.sqlCodePointShift	number	0	負載公用程式的程式SQL碼點偏移。運行移位字符過程。當你的目標數據庫DB2是Postgreql 時需要的。

金鑰	Type	預設值	描述
load.applyRollback	boolean	false	將此參數設定true為，表示您希望服務在將資料載入資料庫時遇到錯誤時回復資料表變更。
forcedDate	string		強制日期提供的日期，如果有一個。
frozenDate	布林值	true	指定是否凍結日期。僅在同時設forcedDate 定時套用。
jcl.type	string	MV	.jcl 檔案類型。允許的值是jcl/vse。如果檔案對於非 vse jcl 而言為空，IDCAMS公用程式PRINT/REPRO命令會傳回 4。
hasGraphic	boolean	false	該INFUTILB實用程序是否需要處理GRAPHICDB2列。
convertGraphicDataToFullWidth	布林值	true	指定是否將圖形資料轉換為全形格式。

gapwalk-cl-command. 戰爭。

這個可選的 Web 應用程序包含 AS/400 實用程序的支持。

此表格提供此應用程式之索引鍵/值參數的詳盡檢視。

金鑰	Type	預設值	描述
logging.config	路徑	類路徑:記錄公用程式 .xml	登入組態檔案參照的標準金鑰。其他標準記錄鍵也可用。
spring.jta.enabled	boolean	false	標準鍵。如果數據源支持模式不是靜態的 xa，則必須禁用彈簧 JTA事務 auto 配置。
spring.datasource.primary.jndi-name	string	jdbc /主要	主要資料來源的JNDI名稱 (Java 命名和目錄介面) (如果使用的話)。JNDI
primary.datasource + -driver-class-name + -url + -username + -password	帶有子鍵的標準彈簧數據源		包含應用程式資料庫的連線資訊 (如果未使用) JNDI。必須具有與現代化應用程式YAML檔案中相同的組態。 或者，強烈鼓勵使用 AWS 秘密，如中the section called “用戶端資料”所述。
encoding	string	ASCII	公用程式中使用的編碼。期望有效的編碼 CP1047IBM930、ASCII、、U
zonedMode	string	EBCDIC_STRICT	編碼或解碼分區資料類型的模式。允許的值是EBCDIC_STRICT /EBCDIC_MODIFIED /AS400。

金鑰	Type	預設值	描述
commands-off	string		要關閉的指令清單，以逗號分隔。允許的值為PGM_BASIC RCVMSG、SNDRCVF、CHGVAR 當您要停用或覆寫現有程式時很有用。PGM_BASIC 是專為調試目的而設計的特定 AWS 藍光時代運行時程序。
forcedDate	string		強制日期提供的日期，如果有一個。

gapwalk-hierarchical-support. 戰爭。

這個可選的 Web 應用程序包含IMS/MFS事務支持。

此表格提供此應用程式之索引鍵/值參數的詳盡檢視。

金鑰	Type	預設值	描述
logging.config	路徑	類路徑:記錄公用程式 .xml	登入組態檔案參照的標準金鑰。其他標準記錄鍵也可用。
spring.jta.enabled	boolean	false	標準鍵。如果數據源支持模式不是靜態的 xa，則必須禁用彈簧 JTA事務 auto 配置。
jhdb.configuration.context.encoding	string		JHDB(Java 階層式資料庫) 編碼。需要有效的編碼字串CP1047IBM930、ASCII、、

金鑰	Type	預設值	描述
jhdb.checkpointPersistence	string	無	檢查點持久性模式。允許的值是none/add/end。用add於在建立新檢查點並將其新增至登錄時保留檢查點。在服務器關閉時使用end太持久檢查點。任何其他值都會停用持續性。請注意，每次將新的檢查點添加到註冊表時，所有現有的檢查點都將被序列化，並且該文件將被刪除。它不是附加到文件中的現有數據。因此，根據檢查站的數量，它可能會對性能產生一些影響。

AWS 藍光時代運行時中可用的 Redis 緩存屬性

您可以使用本文檔來了解 AWS 藍光時代運行時中的 Redis 緩存，以及 Gapwalk 配置，支持的 Redis 屬性以及文application-main.yml件如何引用 Redis 緩存ARN的秘密。

AWS 藍光時代運行時中的 Redis 緩存

Redis 的服務器可以用作 AWS 藍光時代 Gapwalk 應用程序中各種功能的緩存，例如：

AWS 使用 Redis 緩存的藍色時代運行時功能	描述
布魯薩姆緩存	Redis Blusam 快取可使用寫入策略有效讀取記錄，以最佳化批次承載中遇到的寫入密集型工作負載。

AWS 使用 Redis 緩存的藍色時代運行時功能	描述
布魯薩姆鎖	用於資料集和記錄的分散式鎖定的快取。
資料集目錄	目錄資料集快取。
會話緩存	一個 Redis 的快取。 HttpSession緩存存儲用戶名，與 Angular 前端的對話狀態，以及特定的「方言」（BMSMFS，AS400）信息。
會話跟踪器	具有關聯使用者名稱和 session-creation-time資訊的作用中工作階段快取。
JICS快取	JICS資源定義的快取。
TS 佇列	TS 佇列的儲存空間。
JCL檢查點	JCL檢查點緩存。
檔案鎖定	按工作分佈式文件鎖定的緩存。
藍四鎖	用於存儲藍 4iv 記錄鎖。

雷迪斯間隙行走配置

如果`redis`指定為快取機制且未針對特定功能提供 Redis 組態，則會使用全域 Redis 組態。此設定可讓您同時針對多個 Redis 快取使用相同的組態。

在下面的例子中，Blusam 數據集緩存和JICS緩存使用`gapwalk.redis`（`redis.server1`）配置，因為它們的緩存類型設置為`redis`，並且沒有隱式 Redis 屬性在和下指定。the section called “JICS資源定義” the section called “JICS資源定義”但是，Blusam 鎖定緩存將使用不同的 Redis 配置（`redis.server2`），因為它的 Redis 屬性是明確定義的。

```
...

 gapwalk:
   redis:
     hostName: redis.server1
  port: 6379
 ...
```

```
bluesam:
  # Redis bluesam cache
  cache: redis
  # Redis locks cache
  locks:
    cache: redis
 hostName: redis.server2
 port: 6379
...
# Redis jics cache
jics:
  resource-definitions:
    store-type: redis
...
```

若要啟用全域 Redis 配置，請在`main-application.yml`中新增下列組態。

```
gapwalk:
  redis:
    hostName: localhost
    port: 6379
    mode: standalone                        # Optional
    username:                               # Optional
    password: ""                            # Optional
    useSsl: false                           # Optional
    database: 0                             # Optional
    maxTotal: 128                           # Optional
    maxIdle: 128                            # Optional
    minIdle: 16                             # Optional
    testOnBorrow: true                      # Optional
    testOnReturn: true                      # Optional
    testWhileIdle: true                     # Optional
    testOnCreate: true                      # Optional
    minEvictableIdleTimeMillis: 60000       # Optional
    timeBetweenEvictionRunsMillis: 30000    # Optional
    numTestsPerEvictionRun: -1              # Optional
    blockWhenExhausted: true                # Optional
    nettyThreads: 32                        # Optional
    subscriptionsPerConnection: 10          # Optional
    subscriptionConnectionPoolSize: 100     # Optional
    pageSizeInBytes: 8192                   # Optional
```

```
    readTimeout: 2000                    # Optional
```

支持的紅色屬性

下表顯示 AWS 藍光時代執行階段上全域和特定 Redis 快取所支援的 Redis 屬性。

屬性名稱	是否為必要？	描述	值	預設
mode	否	Redis 的運行模式。	standalone | cluster	standalone
hostname	是	Redis 伺服器的主機名稱或 IP 位址。	string	null
port	是	Redis 伺服器偵聽連線的連接埠號碼。	int	null
username	否	驗證的使用者名稱。	string	null
password	否	驗證的密碼。	string	空字符串
useSsl	否	指定是否為 Redis 連接啟用 SSL/TLS加密。	boolean	false
database	否	要使用的 Redis 資料庫編號。Redis 支持多個邏輯數據庫，並且該屬性指定要使用哪一個。	int	0

屬性名稱	是否為必要？	描述	值	預設
maxTotal	否	Redis 連線集區中允許的最大連線數目。	int	128
maxIdle	否	Redis 連線集區中允許的閒置連線數目上限。	int	128
minIdle	否	Redis 連線集區中要維護的閒置連線數目下限。	int	16
testOnBorrow	否	布林值，指出在從集區借用連線之前是否要驗證連線。	布林值	true
testOnReturn	否	一個布爾值，指示是否在將連接返回池之前驗證連接。	布林值	true
testWhileIdle	否	布林值，指出是否定期驗證集區中的閒置連線。	布林值	true
testOnCreate	否	一個布爾值，指示是否在創建連接時驗證。	布林值	true
minEvictableIdleTimeMillis	否	閒置連線必須保留在集區中的最短時間 (以毫秒為單位)，才能收回。	long	600 公升

屬性名稱	是否為必要？	描述	值	預設
timeBetweenEvictionRunsMillis	否	閒置連線收回程式繫線連續執行之間的時間 (以毫秒為單位)。	long	3 億升
numTestsPerEvictionRun	否	閒置連線收回程式執行緒每次執行期間要測試的連線數目上限。	int	-1
blockWhenExhausted	否	Boolean 值，指出當集區耗盡時，是否要封鎖並等待連線變為可用。	布林值	true
nettyThreads	否	用來處理 Redis 連線的網路執行緒數目。	int	32
subscriptionsPerConnection	否	每個 Redis 連線允許的訂閱數目上限。	int	10
subscriptionConnectionPoolSize	否	Redis 訂閱連線集區中允許的最大連線數目。	int	100
pageSizeInBytes	否	Redis 操作的默認頁面大小（以字節為單位）。	long	262144000
readTimeout	否	Redis 作業的讀取逾時 (以毫秒為單位)。	long	2000

。Redis 的緩存屬性。

雷迪斯布魯桑緩存

```
bluesam:
   cache: redis
 # If the following redis properties are not specified gapwalk.redis configuration will
 be used for this cache
   redis:
     hostName: localhost
     port: 6379
     mode: standalone                          # Optional
     username:                                 # Optional
     password: ""                              # Optional
     useSsl: false                             # Optional
     database: 0                               # Optional
     maxTotal: 128                             # Optional
     maxIdle: 128                              # Optional
     minIdle: 16                               # Optional
     testOnBorrow: true                        # Optional
     testOnReturn: true                        # Optional
     testWhileIdle: true                       # Optional
     testOnCreate: true                        # Optional
     minEvictableIdleTimeMillis: 60000         # Optional
     timeBetweenEvictionRunsMillis: 30000      # Optional
     numTestsPerEvictionRun: -1                # Optional
     blockWhenExhausted: true                  # Optional
     nettyThreads: 32                          # Optional
     subscriptionsPerConnection: 10            # Optional
     subscriptionConnectionPoolSize: 100       # Optional
     pageSizeInBytes: 8192                     # Optional
     readTimeout: 2000                         # Optional
```

雷迪斯布魯桑緩存

```
bluesam:
   locks:
     cache: redis
 # If the following redis properties are not specified gapwalk.redis configuration will
 be used for this cache
       hostName: localhost
       port: 6379
       mode: standalone                          # Optional
```

```
      username:                                # Optional
      password: ""                             # Optional
      useSsl: false                            # Optional
      database: 0                              # Optional
      maxTotal: 128                            # Optional
      maxIdle: 128                             # Optional
      minIdle: 16                              # Optional
      testOnBorrow: true                       # Optional
      testOnReturn: true                       # Optional
      testWhileIdle: true                      # Optional
      testOnCreate: true                       # Optional
      minEvictableIdleTimeMillis: 60000        # Optional
      timeBetweenEvictionRunsMillis: 30000     # Optional
      numTestsPerEvictionRun: -1               # Optional
      blockWhenExhausted: true                 # Optional
      nettyThreads: 32                         # Optional
      subscriptionsPerConnection: 10           # Optional
      subscriptionConnectionPoolSize: 100      # Optional
      pageSizeInBytes: 8192                    # Optional
      readTimeout: 2000                        # Optional
```

會話緩存

```
spring:
  session:
    store-type: redis
# If the following redis properties are not specified gapwalk.redis configuration will
be used for this cache
jics:
  redis:
    hostName: localhost
    port: 6379
    mode: standalone                         # Optional
    username:                                # Optional
    password: ""                             # Optional
    useSsl: false                            # Optional
    database: 0                              # Optional
    maxTotal: 128                            # Optional
    maxIdle: 128                             # Optional
    minIdle: 16                              # Optional
    testOnBorrow: true                       # Optional
    testOnReturn: true                       # Optional
```

```
    testWhileIdle: true                     # Optional
    testOnCreate: true                      # Optional
    minEvictableIdleTimeMillis: 60000       # Optional
    timeBetweenEvictionRunsMillis: 30000    # Optional
    numTestsPerEvictionRun: -1              # Optional
    blockWhenExhausted: true                # Optional
    nettyThreads: 32                        # Optional
    subscriptionsPerConnection: 10          # Optional
    subscriptionConnectionPoolSize: 100     # Optional
    pageSizeInBytes: 8192                   # Optional
    readTimeout: 2000                       # Optional
```

JICS資源定義

```
jics:
   resource-definitions:
     store-type: redis
 # If the following redis properties are not specified gapwalk.redis configuration will
 be used for this cache
   redis:
     hostName: localhost
     port: 6379
     mode: standalone                       # Optional
     username:                              # Optional
     password: ""                           # Optional
     useSsl: false                          # Optional
     database: 0                            # Optional
     maxTotal: 128                          # Optional
     maxIdle: 128                           # Optional
     minIdle: 16                            # Optional
     testOnBorrow: true                     # Optional
     testOnReturn: true                     # Optional
     testWhileIdle: true                    # Optional
     testOnCreate: true                     # Optional
     minEvictableIdleTimeMillis: 60000      # Optional
     timeBetweenEvictionRunsMillis: 30000   # Optional
     numTestsPerEvictionRun: -1             # Optional
     blockWhenExhausted: true               # Optional
     nettyThreads: 32                       # Optional
     subscriptionsPerConnection: 10         # Optional
     subscriptionConnectionPoolSize: 100    # Optional
     pageSizeInBytes: 8192                  # Optional
     readTimeout: 2000                      # Optional
```

JICSTS 佇列

```
jics:
   parameters:
     tsqimpl: redis
 # If the following redis properties are not specified gapwalk.redis configuration will
 be used for this cache
   queues:
     ts:
       redis:
         hostName: localhost
         port: 6379
         mode: standalone                          # Optional
         username:                                 # Optional
         password: ""                              # Optional
         useSsl: false                             # Optional
         database: 0                               # Optional
         maxTotal: 128                             # Optional
         maxIdle: 128                              # Optional
         minIdle: 16                               # Optional
         testOnBorrow: true                        # Optional
         testOnReturn: true                        # Optional
         testWhileIdle: true                       # Optional
         testOnCreate: true                        # Optional
         minEvictableIdleTimeMillis: 60000         # Optional
         timeBetweenEvictionRunsMillis: 30000      # Optional
         numTestsPerEvictionRun: -1                # Optional
         blockWhenExhausted: true                  # Optional
         nettyThreads: 32                          # Optional
         subscriptionsPerConnection: 10            # Optional
         subscriptionConnectionPoolSize: 100       # Optional
         pageSizeInBytes: 8192                     # Optional
         readTimeout: 2000                         # Optional
```

會話跟踪器

```
session-tracker:
   store-type: redis
 # If the following redis properties are not specified gapwalk.redis configuration will
 be used for this cache
   redis:
     hostName: localhost
     port: 6379
```

```
    mode: standalone                        # Optional
    username:                               # Optional
    password: ""                            # Optional
    useSsl: false                           # Optional
    database: 0                             # Optional
    maxTotal: 128                           # Optional
    maxIdle: 128                            # Optional
    minIdle: 16                             # Optional
    testOnBorrow: true                      # Optional
    testOnReturn: true                      # Optional
    testWhileIdle: true                     # Optional
    testOnCreate: true                      # Optional
    minEvictableIdleTimeMillis: 60000       # Optional
    timeBetweenEvictionRunsMillis: 30000    # Optional
    numTestsPerEvictionRun: -1              # Optional
    blockWhenExhausted: true                # Optional
    nettyThreads: 32                        # Optional
    subscriptionsPerConnection: 10          # Optional
    subscriptionConnectionPoolSize: 100     # Optional
    pageSizeInBytes: 8192                   # Optional
    readTimeout: 2000                       # Optional
```

JCL檢查點

```
jcl:
   checkpoint:
     provider: redis
 # If the following redis properties are not specified gapwalk.redis configuration will
 be used for this cache
   redis:
     hostname: localhost
     port: 6379
     mode: standalone                       # Optional
     username:                              # Optional
     password: ""                           # Optional
     useSsl: false                          # Optional
     database: 0                            # Optional
     maxTotal: 128                          # Optional
     maxIdle: 128                           # Optional
     minIdle: 16                            # Optional
     testOnBorrow: true                     # Optional
     testOnReturn: true                     # Optional
     testWhileIdle: true                    # Optional
```

```
    testOnCreate: true                        # Optional
    minEvictableIdleTimeMillis: 60000         # Optional
    timeBetweenEvictionRunsMillis: 30000      # Optional
    numTestsPerEvictionRun: -1                # Optional
    blockWhenExhausted: true                  # Optional
    nettyThreads: 32                          # Optional
    subscriptionsPerConnection: 10            # Optional
    subscriptionConnectionPoolSize: 100       # Optional
    pageSizeInBytes: 8192                     # Optional
    readTimeout: 2000                         # Optional
```

檔案鎖定

```
filesLocks:
   enabled: true
   retryTime: 1000
   MaxRetry: 5
   provider: redis
 # If the following redis properties are not specified gapwalk.redis configuration will
 be used for this cache
   redis:
     hostName: localhost
     port: 6379
     mode: standalone                          # Optional
     username:                                 # Optional
     password: ""                              # Optional
     useSsl: false                             # Optional
     database: 0                               # Optional
     pool:
       maxTotal: 128                           # Optional
       maxIdle: 128                            # Optional
       minIdle: 16                             # Optional
       testOnBorrow: true                      # Optional
       testOnReturn: true                      # Optional
       testWhileIdle: true                     # Optional
       testOnCreate: true                      # Optional
       minEvictableIdleTimeMillis: 60000       # Optional
       timeBetweenEvictionRunsMillis: 30000    # Optional
       numTestsPerEvictionRun: -1              # Optional
       blockWhenExhausted: true                # Optional
       nettyThreads: 32                        # Optional
       subscriptionsPerConnection: 10          # Optional
       subscriptionConnectionPoolSize: 100     # Optional
```

```
      pageSizeInBytes: 8192                    # Optional
      readTimeout: 2000                        # Optional
```

藍四鎖

```
blu4iv.lock: redis
blu4iv.lock.timeout: 10 #(in millisecondes)
 # If the following redis properties are not specified gapwalk.redis configuration
will be used for this cache
blu4iv.lock.redis:
      hostName: localhost
      port: 6379
      mode: standalone                         # Optional
      username:                                # Optional
      password: ""                             # Optional
      useSsl: false                            # Optional
      database: 0                              # Optional
      maxTotal: 128                            # Optional
      maxIdle: 128                             # Optional
      minIdle: 16                              # Optional
      testOnBorrow: true                       # Optional
      testOnReturn: true                       # Optional
      testWhileIdle: true                      # Optional
      testOnCreate: true                       # Optional
      minEvictableIdleTimeMillis: 60000        # Optional
      timeBetweenEvictionRunsMillis: 30000     # Optional
      numTestsPerEvictionRun: -1               # Optional
      blockWhenExhausted: true                 # Optional
      nettyThreads: 32                         # Optional
      subscriptionsPerConnection: 10           # Optional
      subscriptionConnectionPoolSize: 100      # Optional
      pageSizeInBytes: 8192                    # Optional
      readTimeout: 2000                        # Optional
```

資料集目錄

```
datasimplifier:
   catalogImplementation: redis
   # If the following redis properties are not specified gapwalk.redis configuration
 will be used for this cache
   redis:
     hostName: localhost
```

```
    port: 6379
    mode: standalone                          # Optional
    username:                                 # Optional
    password: ""                              # Optional
    useSsl: false                             # Optional
    database: 0                               # Optional
    maxTotal: 128                             # Optional
    maxIdle: 128                              # Optional
    minIdle: 16                               # Optional
    testOnBorrow: true                        # Optional
    testOnReturn: true                        # Optional
    testWhileIdle: true                       # Optional
    testOnCreate: true                        # Optional
    minEvictableIdleTimeMillis: 60000         # Optional
    timeBetweenEvictionRunsMillis: 30000      # Optional
    numTestsPerEvictionRun: -1                # Optional
    blockWhenExhausted: true                  # Optional
    nettyThreads: 32                          # Optional
    subscriptionsPerConnection: 10            # Optional
    subscriptionConnectionPoolSize: 100       # Optional
    pageSizeInBytes: 8192                     # Optional
    readTimeout: 2000                         # Optional
```

秘密經理 Redis 的緩存

該`application-main.yaml`文件可以引ARN用 Redis 緩存的秘密。如需如何整合 AWS Secrets Manager 以在執行階段安全擷取 Redis 連線詳細資料的詳細資訊，請參閱the section called “AWS 藍光時代運行時秘密”。

設定 Gapwalk 應用程式的安全性

下列主題說明如何保護 Gapwalk 應用程式的安全。

您有責任提供正確的配置，以確保 AWS Blu Age 框架的使用是安全的。

依預設，會停用所有與安全性相關的功能。若要啟用驗證 (和CSRFXSSCSP、、等等)，`gapwalk-application.security`請將設定`gapwalk-application.security.identity`為`enabled`和`oauth`。

主題

- 設定 Gapwalk 應用程式的URI可存取

- 設定 Gapwalk 應用程式的驗證

設定 Gapwalk 應用程式的URI可存取

本主題說明如何設定 Gapwalk 應用程式的URIs篩選。此功能不需要身分識別提供者 (IdP)。

若要封鎖清單URIs，請將下列兩行新增至現代化應用程式，並取application-main.yml代 *URI-1*, *URI-2*，等等，與你想要阻止。URIs

```
gapwalk-application.security.filterURIs: enabled
gapwalk-application.security.blockedURIs: URI-1, URI-2, URI-3
```

設定 Gapwalk 應用程式的驗證

若要設定 Gapwalk 應用程式的OAuth2驗證，您需要設定身分識別提供者 (IdP)，並將其與您的應用程式整合。本指南涵蓋使用 Amazon Cognito 或鍵盤斗篷作為 IdP 的步驟。使用 Amazon Cognito，您可以使用 Cognito 使用者集區詳細資料更新應用程式的組態檔案。使用 Keycloak，您可以根據使用者指派的角色來控制對應用程式APIs和資源的存取。

主題

- 使用亞馬遜認可設定 Gapwalk OAuth2 身份驗證
- 使用鍵盤遮罩設定 Gapwalk OAuth2 驗證

使用亞馬遜認可設定 Gapwalk OAuth2 身份驗證

本主題說明如何使用 Amazon Cognito 做為身分識別提供者 (IdP)，為 Gapwalk 應用程式設定身分OAuth2驗證。

必要條件

在本教程中，我們將使用 Amazon Cognito 作為 IdP 和 PlanetDemo 現代化項目。

您可以使用任何其他外部身分識別提供者。這些資 ClientRegistration 訊必須從您的 IdP 取得，並且是 Gapwalk 驗證所必需的資訊。如需詳細資訊，請參閱 Amazon Cognito 開發人員指南。

信 ClientRegistration 息：

客戶端標識

的識別碼 ClientRegistration。在我們的例子中，它將是 PlanetsDemo。

用戶端機密

您的客戶密碼。

授權端點

授權伺服器URI的授權端點。

令牌端點

授權伺服器URI的權杖端點。

JWKS 端點

URI用於獲取 JSON Web Key（JWK），該密鑰包含用於驗證授權服務器發出的 JSON Web 簽名的密鑰。

重定向 URI

如果授與存URI取權，授權伺服器會將終端使用者重新導向的目標。

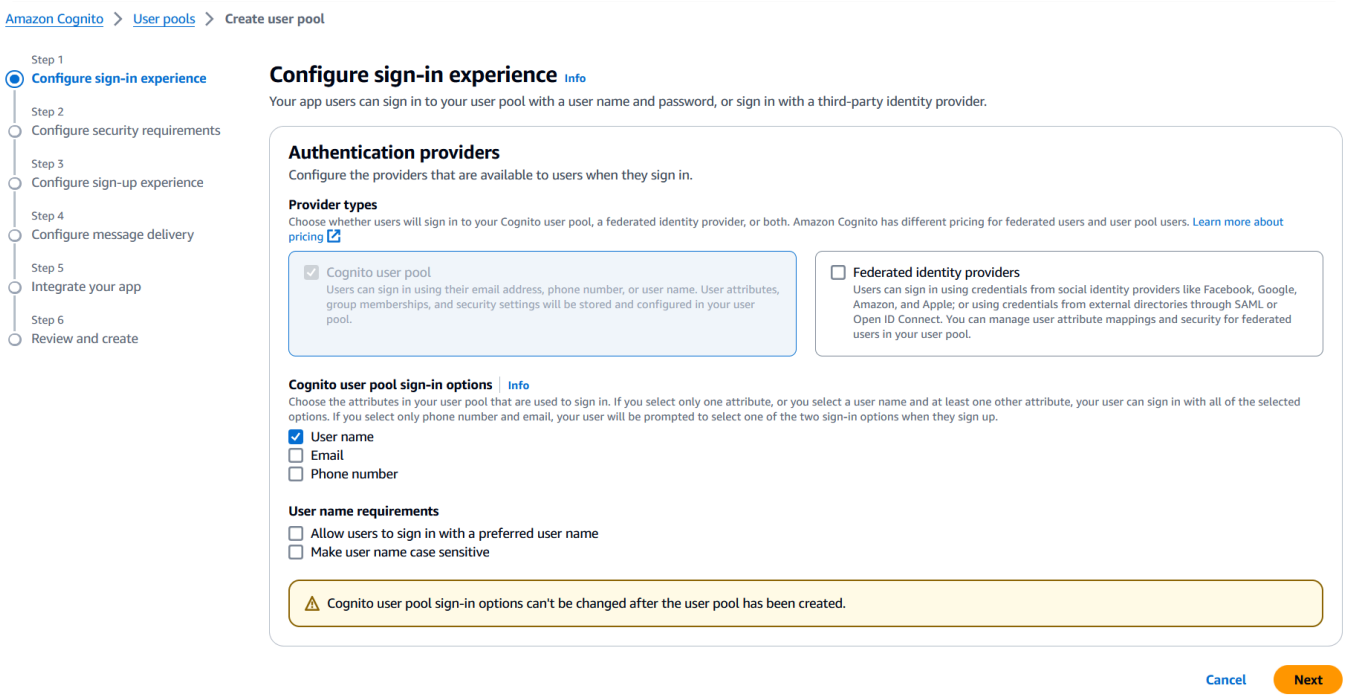
Amazon Cognito 設置

首先，我們將建立並設定 Amazon Cognito 使用者集區和使用者，以便與部署的 Gapwalk 應用程式搭配使用以進行測試。

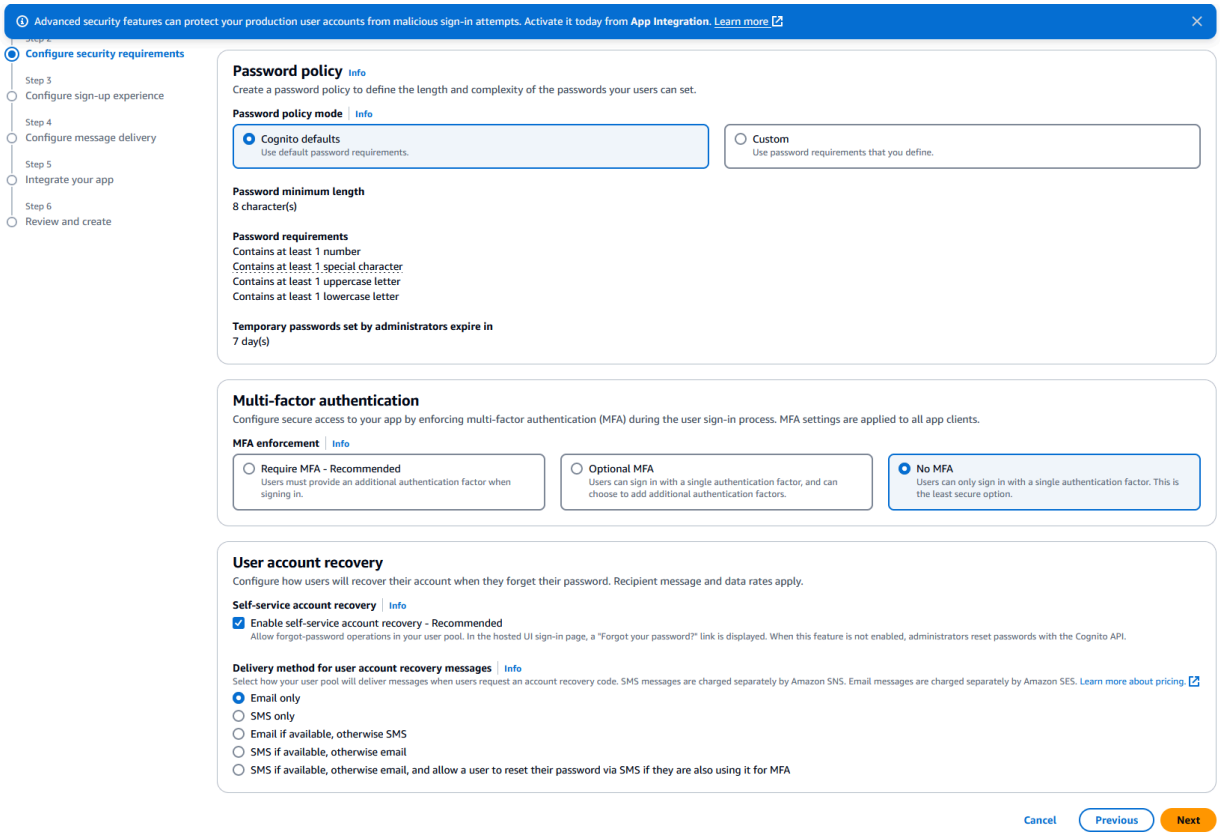
> **Note**
>
> 如果您正在使用其他 IdP，則可以略過此步驟。

建立使用者集區

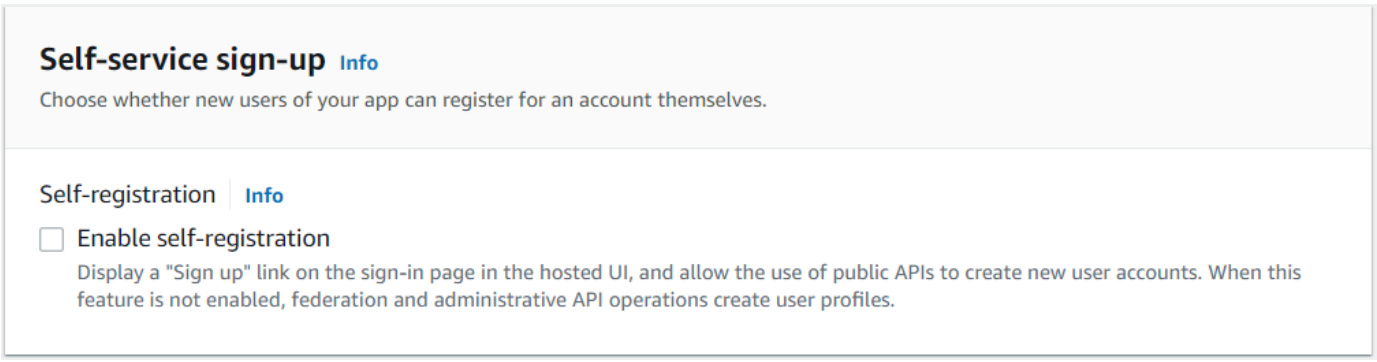
1. 轉到 Amazon Cognito，然後使用您的 AWS 憑據 AWS Management Console 進行身份驗證。
2. 選擇 User Pools (使用者集區)。
3. 選擇 Create a user pool (建立使用者集區)。
4. 在 [設定登入體驗] 中，保留 Cognito 使用者集區預設提供者類型。您可以選擇一或多個 Cognito 使用者集區登入選項；目前請選擇 [使用者名稱]，然後選擇 [下一步]。



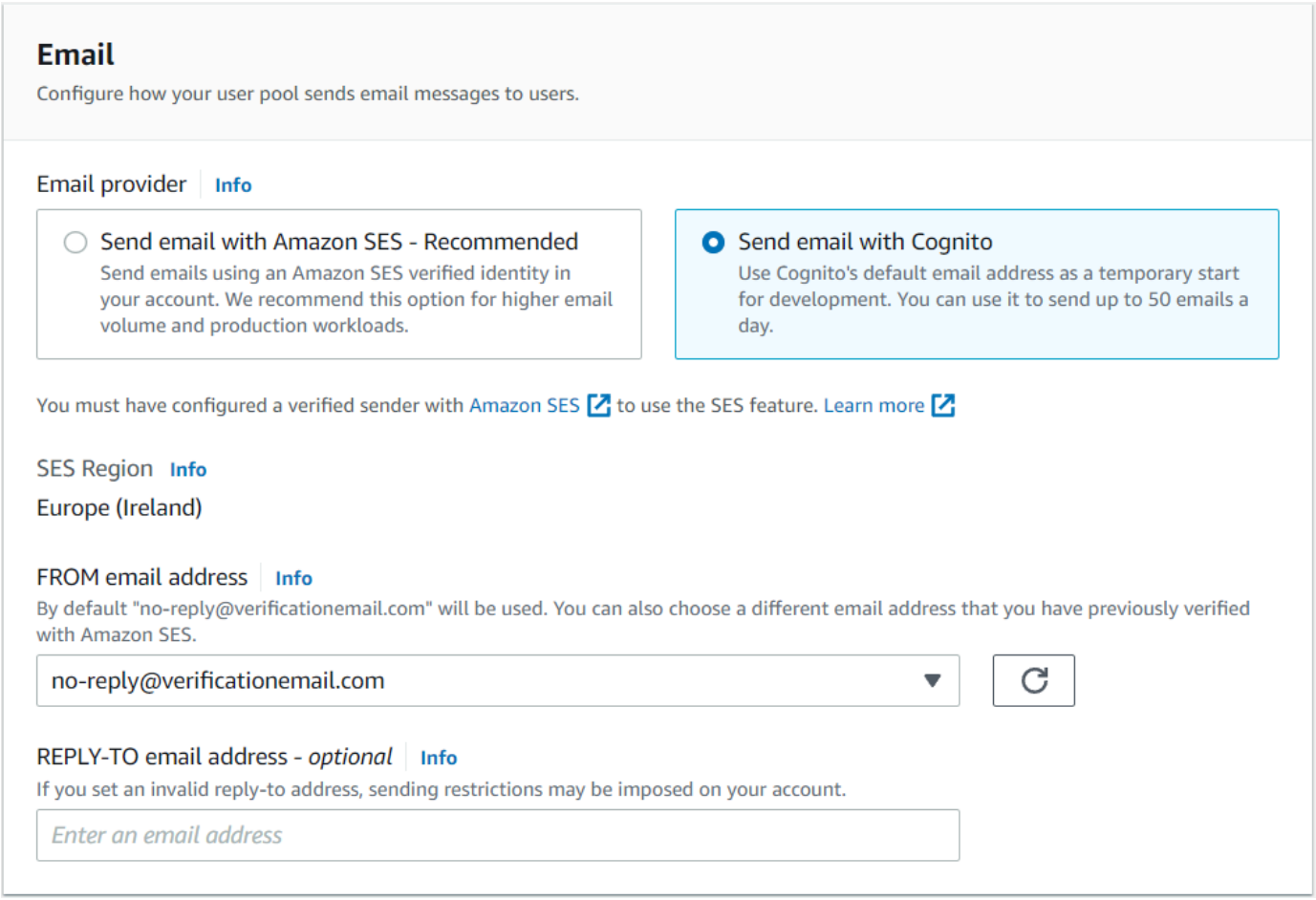
5. 在 [設定安全性需求] 中，保留預設值並選擇否停用多重要素驗證MFA，然後選擇 [下一步]。



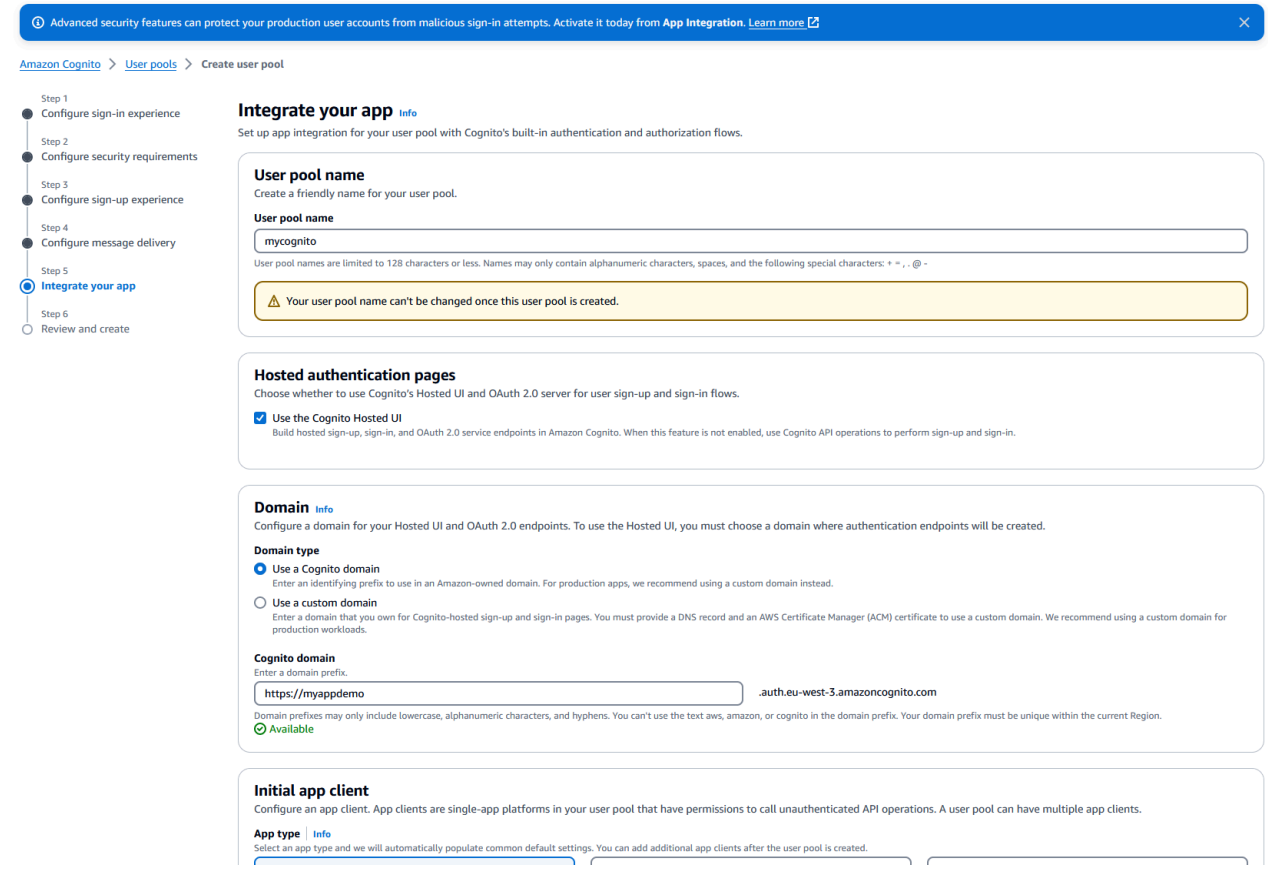
6. 基於安全性考量，請停用 [啟用自我註冊]，然後選擇 [下一步]。



7. 選擇使用 Cognito 傳送電子郵件，然後選擇 [下一步]。



8. 在 [整合您的應用程式] 中，指定使用者集區的名稱。在託管驗證頁面中，選擇「使用 Cognito 託管的 UI」。



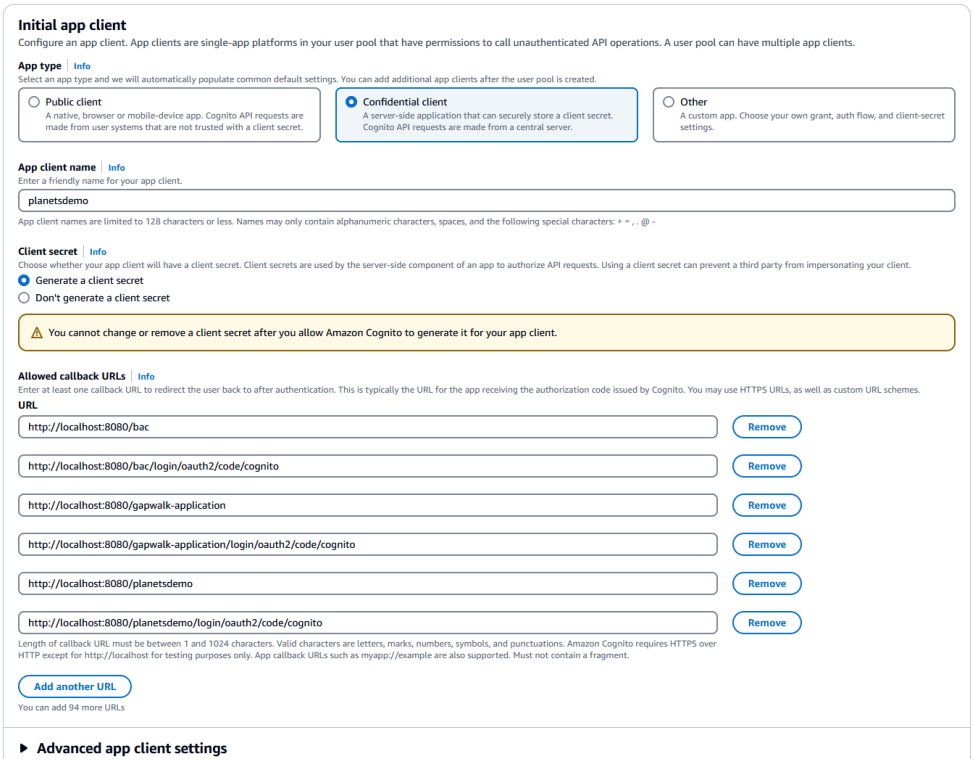
9. 為了簡化起見，請在「網域」中選擇「使用 Cognito 網域」，然後輸入網域前置字元，`https://planetsdemo`例如。演示應用程序必須添加為客戶端。

   a. 在初始應用程式用戶端中，選擇機密用戶端 輸入應用程式用戶端名稱，例如**`planetsdemo`**，然後選擇 [產生用戶端密碼]。

   b. 在 [允許的回呼] 中，URL輸入URL要在驗證後將使用者重新導向至。必URL須以結束/`login/oauth2/code/cognito`。例如，對於我們的應用程序和後端 Gapwalk 和BAC應用程序：

```
http://localhost:8080/bac
      http://localhost:8080/bac/login/oauth2/code/cognito
      http://localhost:8080/gapwalk-application
      http://localhost:8080/gapwalk-application/login/oauth2/code/cognito
      http://localhost:8080/planetsdemo
      http://localhost:8080/planetsdemo/login/oauth2/code/cognito
```

   您可以編輯URL稍後的。



c. 在允許登出中，URLs輸入您希望 Amazon Cognito 在應用程式登出使用者時重新導向至URL的登出頁面。例如，對於後端 Gapwalk 和BAC應用程序：

```
http://localhost:8080/bac/logout
http://localhost:8080/gapwalk-application/logout
http://localhost:8080/planetsdemo/logout
```

您可以編輯URL稍後的。

d. 將預設值保留在 [進階應用程式用戶端設定] 和 [屬性讀取和寫入權限] 區段中。

e. 選擇 Next (下一步)。

10. 在 [檢閱並建立] 中，確認您的選擇，然後選擇 [建立使用者集區]。

如需詳細資訊，請參閱建立使用者集區。

使用者建立

由於自我註冊已停用，因此請建立 Amazon Cognito 使用者。導航到 Amazon Cognito 在 AWS Management Console. 選擇您建立的使用者集區，然後在 [使用者] 中選擇 [建立使用者]。

在 [使用者資訊] 中，選擇 [傳送電子郵件邀請]，輸入使用者名稱和電子郵件地址，然後選擇 [產生密碼]。選擇 Create user (建立使用者)。

角色建立

在「群組」(Groups) 標籤中，建立 3 個群組 (SUPER_ ADMIN ADMIN、、和USER)，並將您的使用者與其中一個或多個群組相關聯。這些角色稍後會USER由 Gapwalk 應用程式對應至 ROLE SUPER ROLE ROLE _ ADMIN、_ ADMIN 和 _，以便能夠存取某些受限制的APIREST呼叫。

將 Amazon Cognito 集成到 Gapwalk 應用程序

現在您的 Amazon Cognito 使用者集區和使用者已準備就緒，請前往現代化應用程式的`application-main.yml`檔案，然後新增下列程式碼：

```
gapwalk-application.security: enabled
gapwalk-application.security.identity: oauth
gapwalk-application.security.issuerUri: https://cognito-idp.<region-id>.amazonaws.com/
<pool-id>
gapwalk-application.security.domainName: <your-cognito-domain>
gapwalk-application.security.localhostWhitelistingEnabled: false

spring:
  security:
    oauth2:
      client:
        registration:
          cognito:
            client-id: <client-id>
            client-name: <client-name>
            client-secret: <client-secret>
            provider: cognito
            authorization-grant-type: authorization_code
            scope: openid
            redirect-uri: "<redirect-uri>"
        provider:
          cognito:
            issuer-uri: ${gapwalk-application.security.issuerUri}
            authorization-uri: ${gapwalk-application.security.domainName}/oauth2/
authorize
            jwk-set-uri: ${gapwalk-application.security.issuerUri}/.well-known/
jwks.json
            token-uri: ${gapwalk-application.security.domainName}/oauth2/token
            user-name-attribute: username
      resourceserver:
        jwt:
          jwk-set-uri: ${gapwalk-application.security.issuerUri}/.well-known/jwks.json
```

如所述取代下列預留位置：

1. 轉到 Amazon Cognito，然後使用您的 AWS 憑據 AWS Management Console 進行身份驗證。
2. 選擇「用戶池」，然后選擇您創建的用戶池。你可以找到 *pool-id* 在使用者集區 ID 中。
3. 選擇應用程序集成，您可以在其中找到 *your-cognito-domain*，然後轉到「應用程序客戶端和分析」，然後選擇您的應用程序。
4. 在應用程式用戶端中：yourApp您可以找到 *client-name*，*client-id* 和 *client-secret* (顯示用戶端密碼)。
5. *region-id* 對應於您在其中建立 Amazon Cognito 使用者和使用者集 AWS 區的區域 ID。範例：`eu-west-3`。
6. 用於 *redirect-uri* 輸入您URI為允許的回呼指定的URL。在我們的例子中，它是`http://localhost:8080/planetsdemo/login/oauth2/code/cognito`。

您現在可以部署 Gapwalk 應用程式，並使用先前建立的使用者登入您的應用程式。

使用鍵盤遮罩設定 Gapwalk OAuth2 驗證

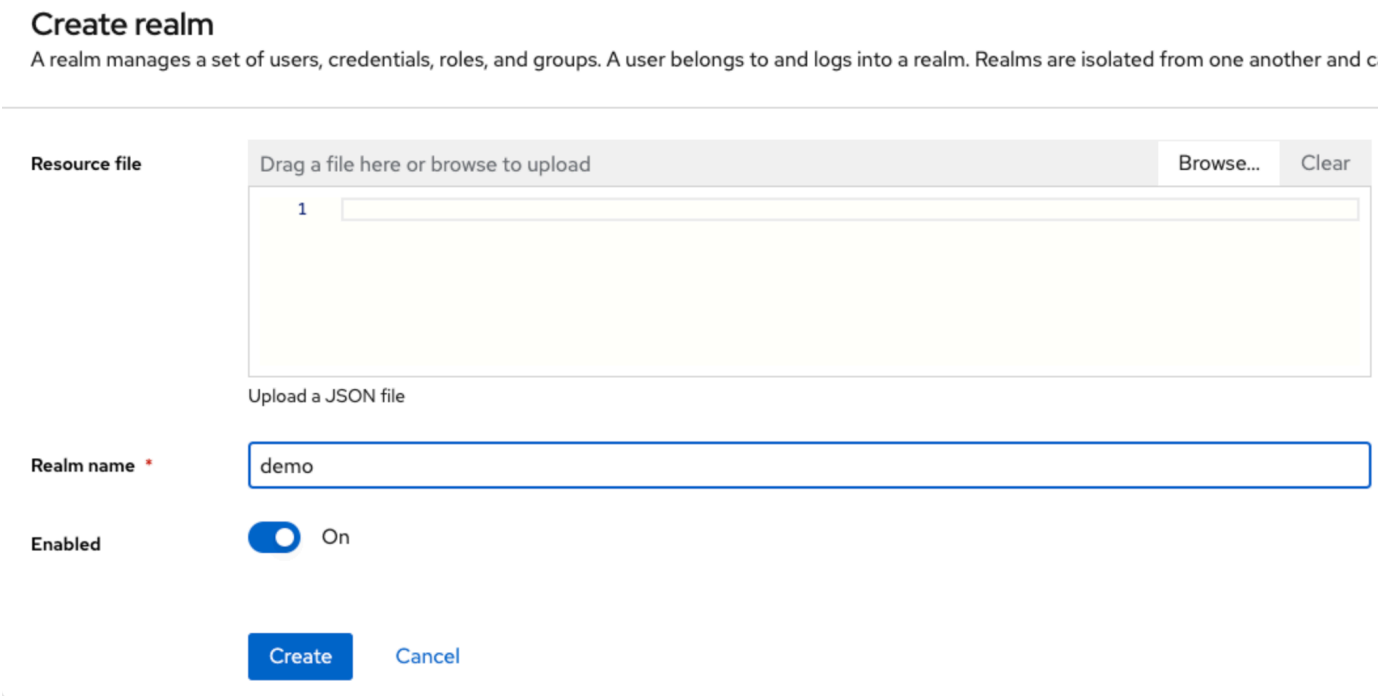
本主題說明如何使用 Keycloak 做為身分識別提供者 (IdP)，為 Gapwalk 應用程式設定OAuth2驗證。在本教程中，我們使用鍵盤斗篷 24.0.0。

必要條件

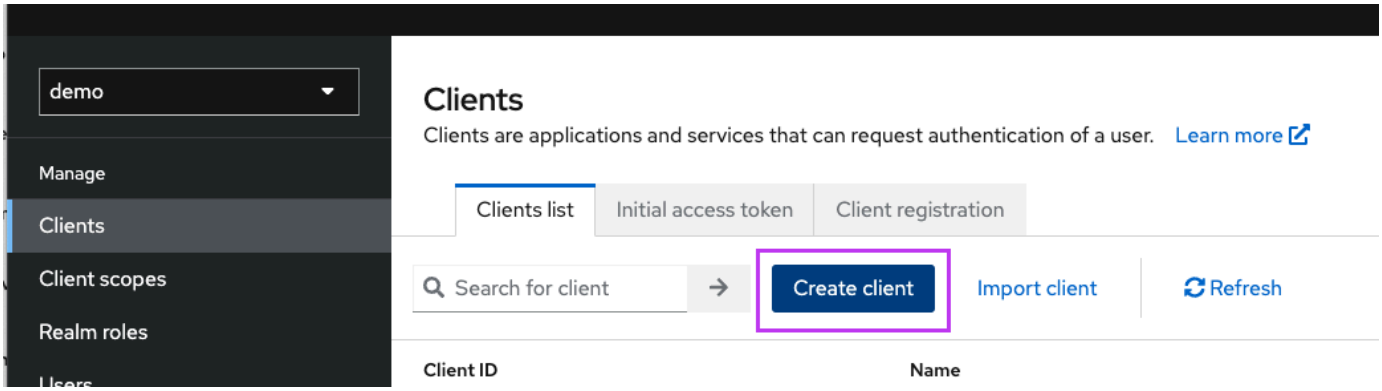
- 鑰匙斗篷
- 間隙應用程序

鑰匙斗篷設置

1. 前往網頁瀏覽器中的 Keycloak 儀表板。預設認證為系統管理員/管理員。移至左上方的導覽列，並使用名稱建立領域**demo**，如下圖所示。



2. 使用名稱建立用戶端**app-demo**。



替換為您的 Gapwalk 應`localhost:8080`用程序的地址

General settings

Client ID * app-demo

Name

Description

Always display in UI Off

Access settings

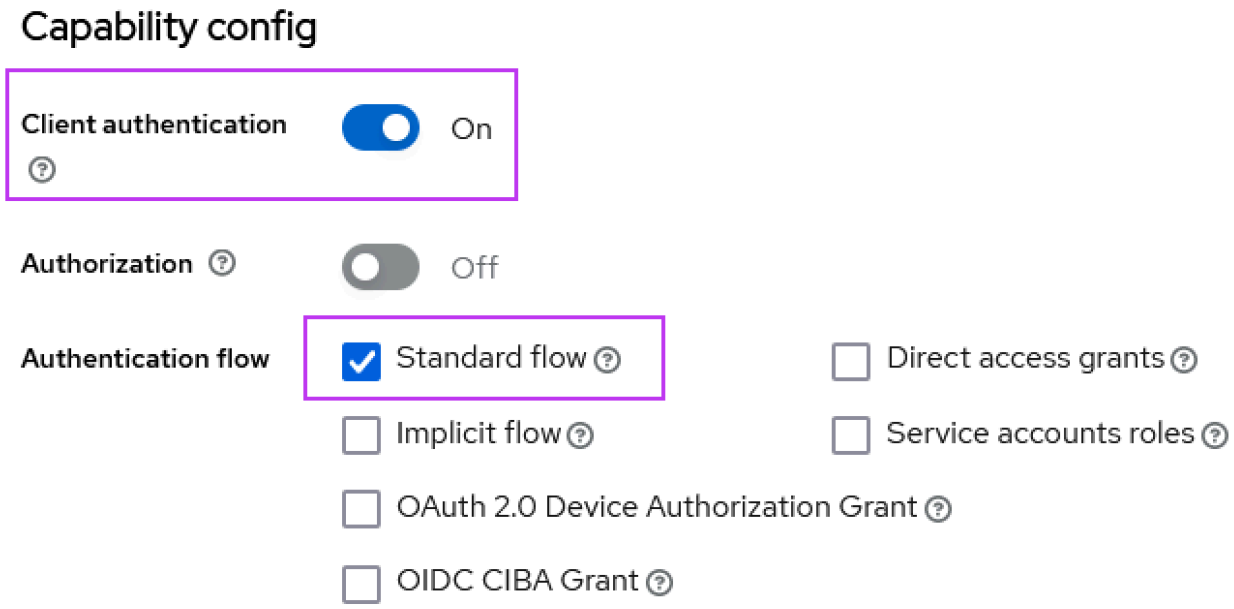
Root URL http://localhost:8080

Home URL

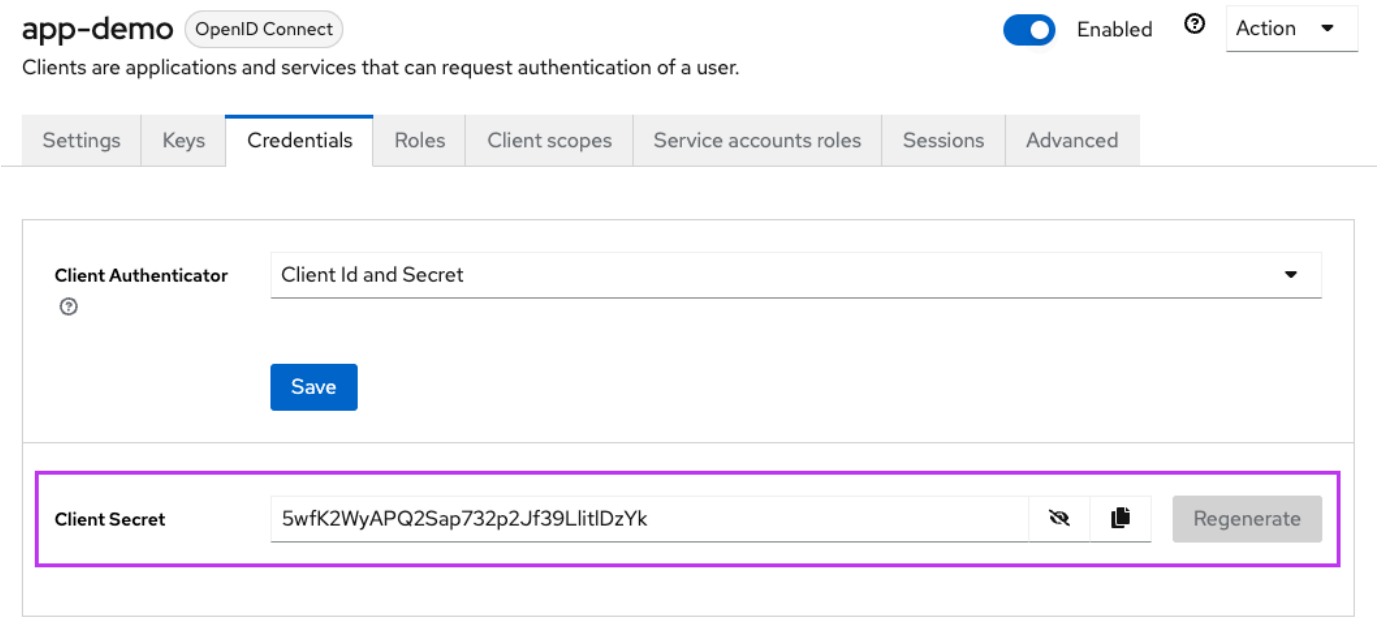
Valid redirect URIs
- http://localhost:8080/*
- https://localhost:8080/*
- Add valid redirect URIs

Valid post logout redirect URIs
- http://localhost:8080/*
- https://localhost:8080/*
- Add valid post logout redirect URIs

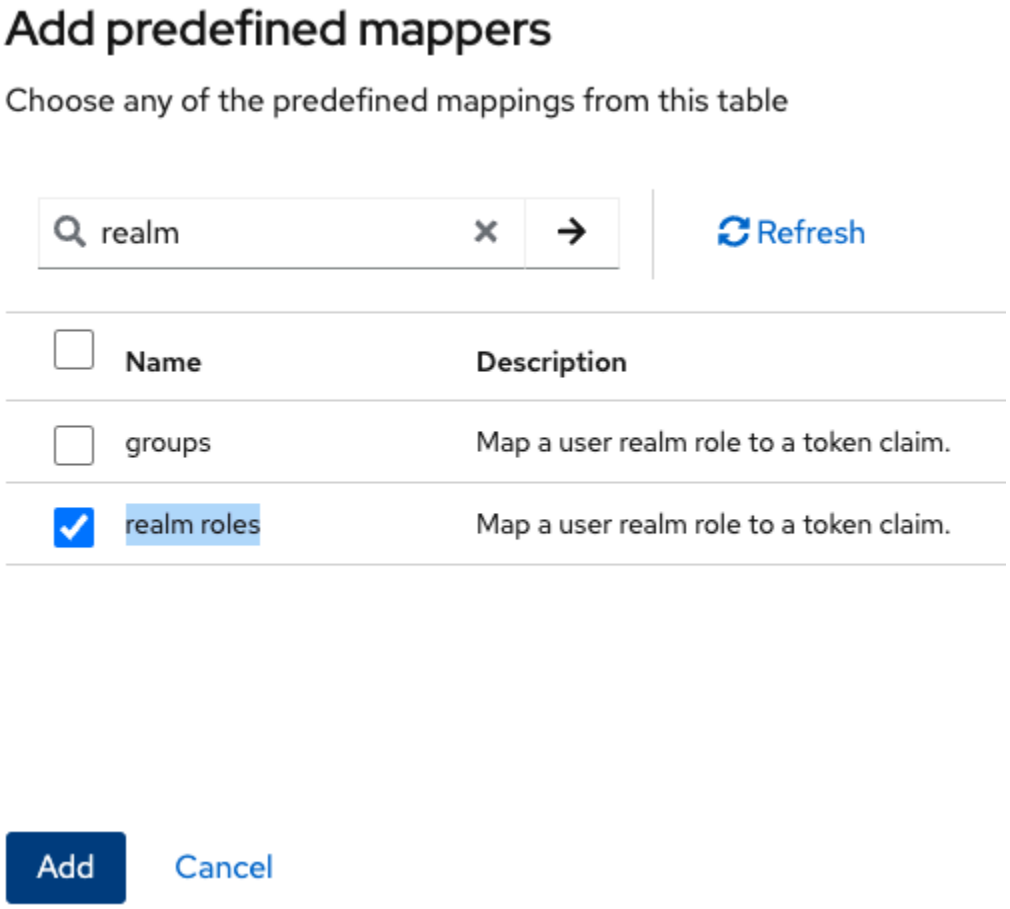
Web origins
- +
- Add web origins



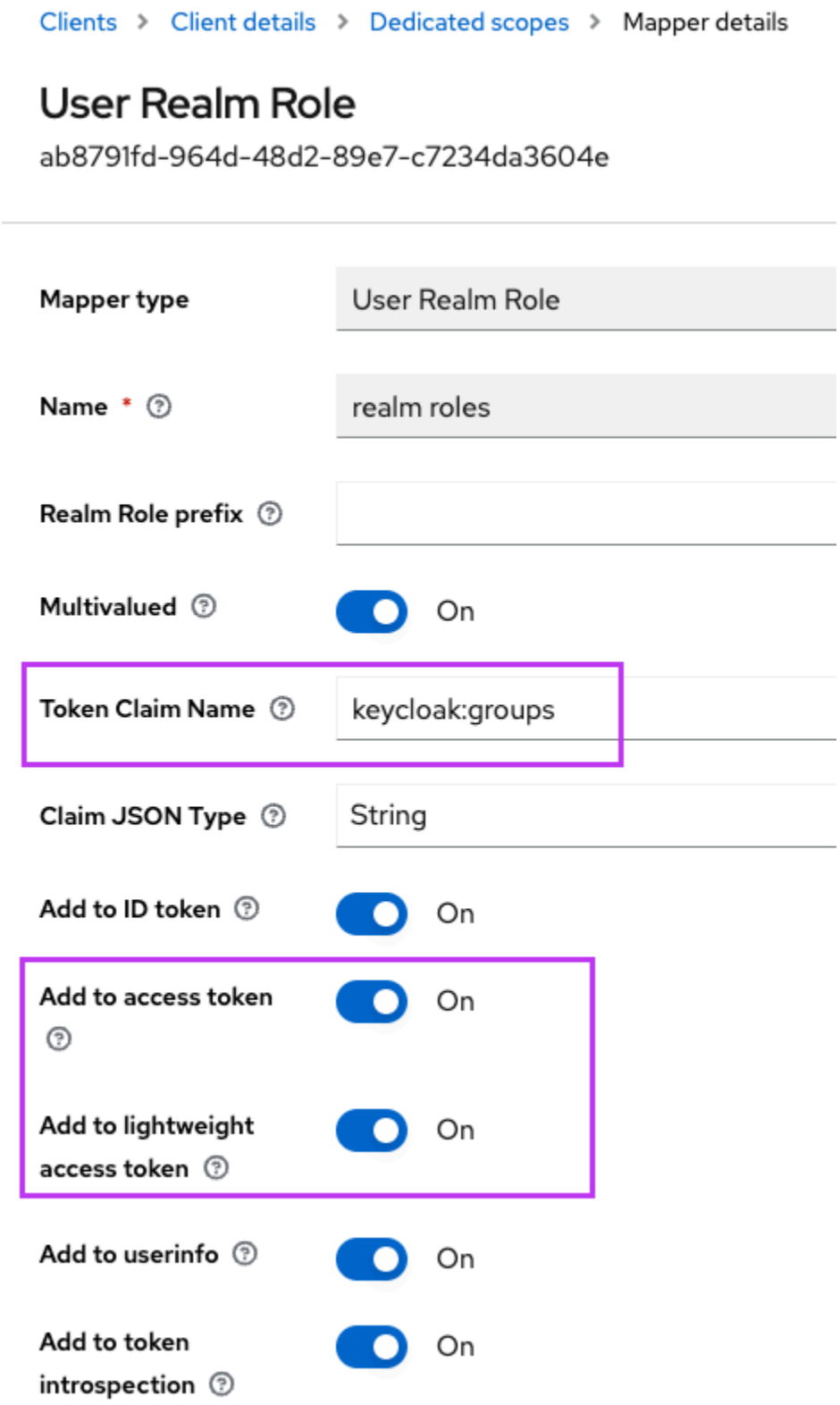
3. 若要取得用戶端密碼，請選擇用戶端，然後選擇應用程式示範，然後選擇認證。



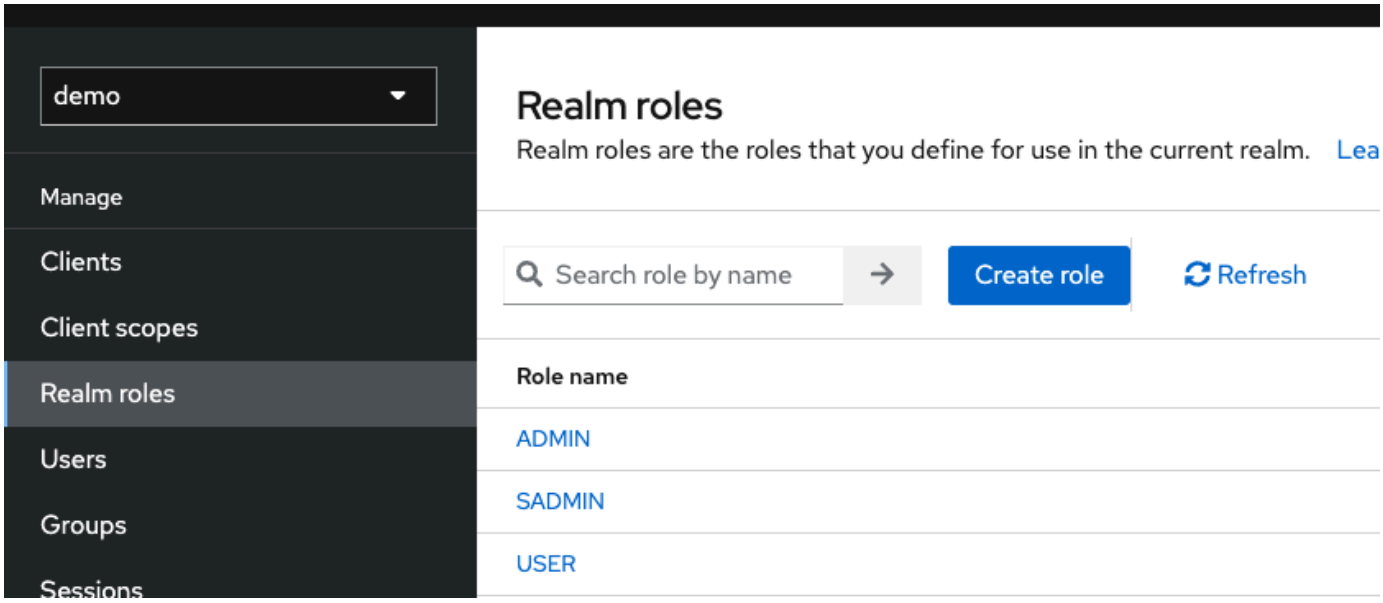
4. 選擇用戶端、用戶端範圍，然後選擇新增預先定義的對應程式 選擇範圍角色。



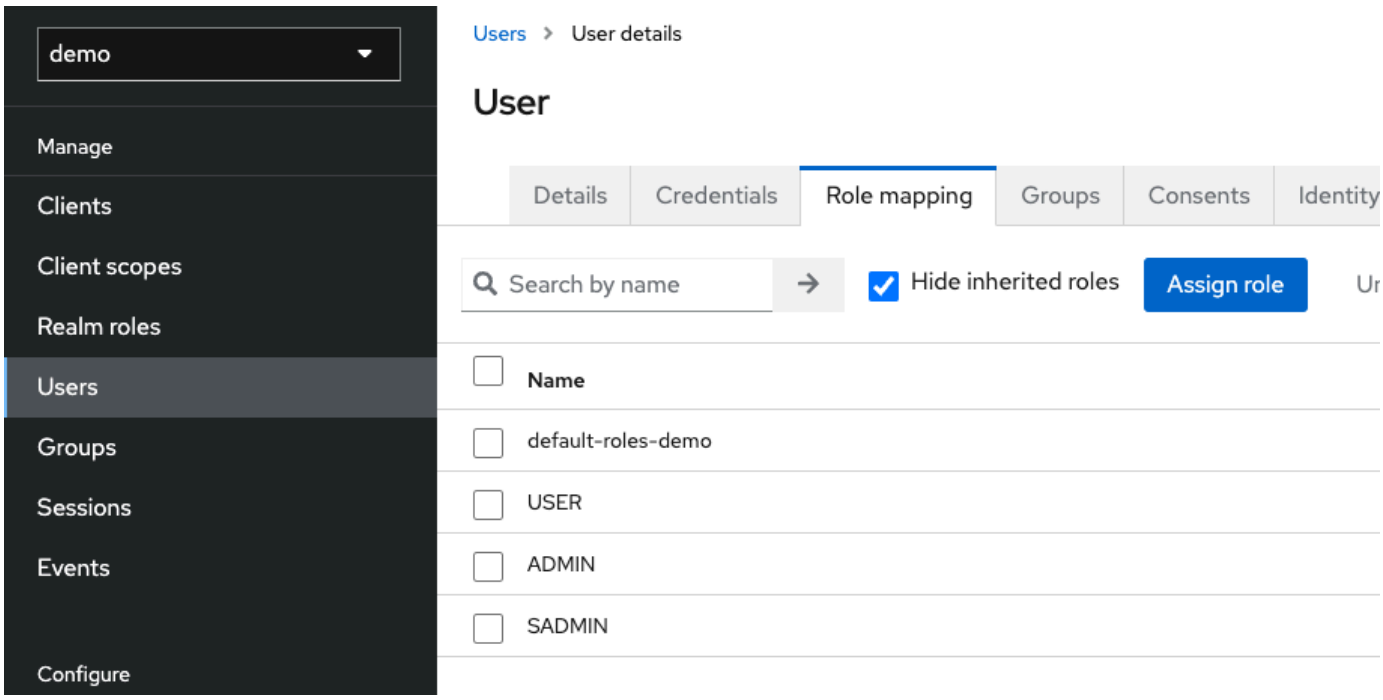
5. 使用下圖所示的組態編輯您的範圍角色。



6. 記住定義的令牌聲明名稱。您需要在`gapwalk-application.security.claimGroupName`性質的 Gapwalk 設定定義中使用此值。



7. 選擇範圍角色，然後建立 3 個角色：**SUPER_ADMINADMIN**、和**USER**。這些角色稍後會ROLE_USER由 Gapwalk 應用程式對應至ROLE_SUPER_ADMINROLE_ADMIN、和，以便能夠存取某些受限制的APIREST呼叫。



將鑰匙斗篷集成到 Gapwalk 應用程序

編輯你的application-main.yml如下：

```
gapwalk-application.security: enabled
```

```
gapwalk-application.security.identity: oauth
gapwalk-application.security.issuerUri: http://<KEYCLOAK_SERVER_HOSTNAME>/realms/
<YOUR_REALM_NAME>
gapwalk-application.security.claimGroupName: "keycloak:groups"

gapwalk-application.security.userAttributeName: "preferred_username"
# Use "username" for cognito,
#     "preferred_username" for keycloak
#      or any other string
gapwalk-application.security.localhostWhitelistingEnabled: false

spring:
  security:
    oauth2:
      client:
        registration:
          demo:
            client-id: <YOUR_CLIENT_ID>
            client-name: Demo App
            client-secret: <YOUR_CLIENT_SECRET>
            provider: keycloak
            authorization-grant-type: authorization_code
            scope: openid
            redirect-uri: "{baseUrl}/login/oauth2/code/{registrationId}"
        provider:
          keycloak:
            issuer-uri: ${gapwalk-application.security.issuerUri}
            authorization-uri: ${gapwalk-application.security.issuerUri}/protocol/
openid-connect/auth
            jwk-set-uri: ${gapwalk-application.security.issuerUri}/protocol/openid-
connect/certs
            token-uri: ${gapwalk-application.security.issuerUri}/protocol/openid-
connect/token
            user-name-attribute: ${gapwalk-application.security.userAttributeName}
      resourceserver:
        jwt:
          jwk-set-uri: ${gapwalk-application.security.issuerUri}/protocol/openid-
connect/certs
```

Replace (取代) *<KEYCLOAK_SERVER_HOSTNAME>*, *<YOUR_REALM_NAME>*, *<YOUR_CLIENT_ID>* 和 *<YOUR_CLIENT_SECRET>* 使用您的 Keycloak 伺服器主機名稱、領域名稱、用戶端 ID 和用戶端密碼。

AWS 藍色時代運行時 APIs

AWS Blu Age Runtime 使用多個 Web 應用程序來公開REST端點，提供了使用REST客戶端與現代化應用程序進行交互的方法（例如使用調度程序調用作業）。

本文件的目的是列出可用的REST端點，並提供有關以下內容的詳細資訊：

- 他們的角色
- 正確使用它們的方法

根據所提供服務的性質以及暴露端點的 Web 應用程式，列出的端點會分為類別。

我們假設您已經擁有使用專用工具（例如 ThunderCli ent POSTMAN，Web 瀏覽器等）使用REST端點的基本知識。CURL或者編寫自己的一段代碼進行API調用。

主題

- 建置時可供使用者使用的端點 URLs
- AWS 藍光時代的 Gapwalk 應用程序的端點
- Blusam 應用程式主控台端點 REST
- 在 AWS 藍光時代管理JICS應用程序控制
- AWS 藍光時代用戶的數據結構

建置時可供使用者使用的端點 URLs

本主題列出端點的URLs具有根路徑。下面的每個 Web 應用程序都定義了一個由所有端點共享的根路徑。然後，每個端點都會新增自己的專用路徑。使用URL的結果是路徑串連的結果。例如，考慮到 Gapwalk 應用程序的第一個端點，我們有：

- `/gapwalk-application`針對根 Web 應用程式路徑。
- `/scripts`用於專用端點路徑。

由此產生URL的使用將是 `http://server:port/gapwalk-application/scripts`

伺服器

指向服務器名稱（託管給定 Web 應用程序的名稱）。

port

伺服器公開的連接埠。

AWS 藍光時代的 Gapwalk 應用程序的端點

在本主題中，您將瞭解 Gapwalk 網路應用程式的端點。這些使用根路徑/gapwalk-application。

主題

- Batch 工作 (現代化JCLs和類似) 相關端點
- 度量端點
- 其他端點
- Job 佇列相關端點

Batch 工作 (現代化JCLs和類似) 相關端點

Batch 工作可以同步或非同步執行 (請參閱下方的詳細資訊)。Batch 作業正在使用 groovy 腳本執行，這些腳本是舊腳本（JCL）的現代化的結果。

主題

- 列出部署的腳本
- 同步啟動指令碼
- 以非同步方式啟動指令碼
- 列出觸發的腳本
- 擷取工作執行細節
- 列出可以殺死的異步啟動腳本
- 列出可以殺死的同步啟動腳本
- 殺死給定的工作執行
- 列出現有的檢查點以進行重新啟動
- 重新啟動工作 (同步)
- 重新啟動工作 (非同步)
- 設定非同步工作執行的執行緒限制

列出部署的腳本

- 支持的方法：GET
- 路徑：`/scripts`
- 引數：無
- 此端點返回服務器上部署的 groovy 腳本的列表，作為一個字符串。此端點主要用於從 Web 瀏覽器中使用，因為結果字符串是一個帶有活動鏈接的HTML頁面（每個可啟動腳本的鏈接-請參閱下面的示例）。

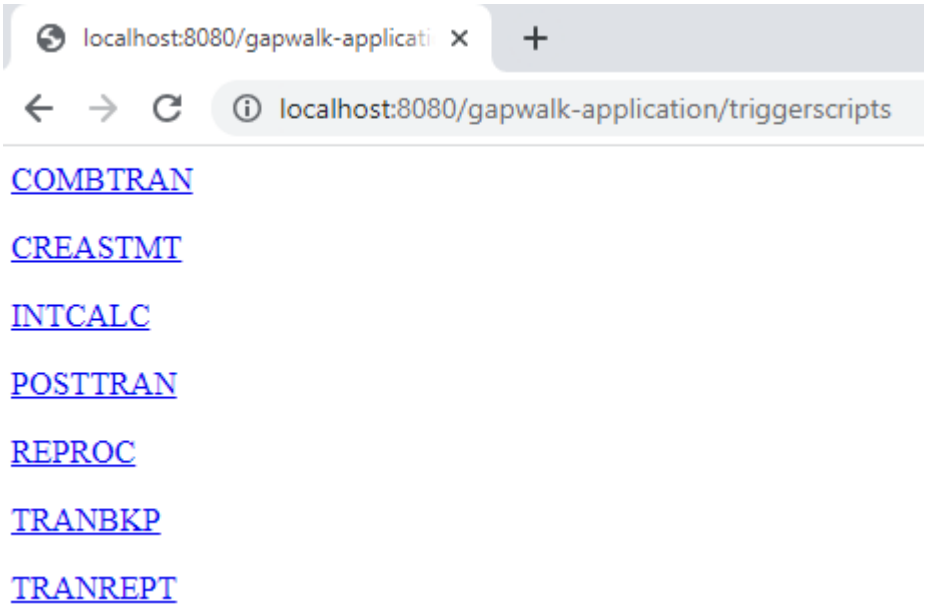
回應範例：

```
<p><a href=./script/COMBTRAN>COMBTRAN</a></p><p><a href=./script/CREASTMT>CREASTMT</
a></p><p><a href=./script/INTCALC>INTCALC</a></p><p><a href=./script/
POSTTRAN>POSTTRAN</a></p><p><a href=./script/REPROC>REPROC</a></p><p><a href=./script/
TRANBKP>TRANBKP</a></p><p><a href=./script/TRANREPT>TRANREPT</a></p><p><a href=./
script/functions>functions</a></p>
```

> **Note**
>
> 這些鏈接表示用於同步啟動每個列出的腳本的 URL。

- 支持的方法：GET
- 路徑：`/triggerscripts`
- 引數：無
- 此端點返回服務器上部署的 groovy 腳本的列表，作為一個字符串。此端點主要用於從 Web 瀏覽器中使用，因為結果字符串是一個帶有活動鏈接的HTML頁面（每個可啟動腳本的鏈接-請參閱下面的示例）。

  與先前的端點響應相反，這些鏈接代表用於異步啟動每個列出的腳本的 URL。



同步啟動指令碼

此端點有兩個變體，具有用於GET和POST使用的專用路徑（請參閱下文）。

- 支持的方法：GET
- 路徑：`/script/{scriptId:.+}`
- 支持的方法：POST
- 路徑：`/post/script/{scriptId:.+}`
- 引數：
  - 要啟動之指令碼的識別碼
  - 可選：使用請求參數傳遞給腳本的參數（視為 `aMap<String,String>`）。給定的參數將被自動添加到調用 groovy 腳本的綁定。
- 該調用將使用給定的標識符啟動腳本，如果提供的話使用額外的參數，並等待腳本執行完成，然後返回一個 message (`String`)，該消息將是：
  - 「完成。」（如果作業執行順利）。
  - JSON錯誤訊息，其中包含工作執行期間發生錯誤的詳細資訊。您可以從伺服器記錄擷取進一步的詳細資訊，以瞭解工作執行出了什麼問題。

```
{
    "exitCode": -1,
    "stepName": "STEP15",
```

```
    "program": "CBACT04C",
    "status": "Error"
}
```

查看服務器日誌，我們可以發現這是一個部署問題（預期的程序尚未正確部署，因此無法找到，導致作業執行失敗）：

```
2023-06-09 10:27:28 default        INFO  - c.n.b.g.r.s.BatchWebController          - --> executing script INTCALC
2023-06-09 10:27:28 default        INFO  - c.n.b.g.r.s.BatchWebController          - Bound jobContext 419695287 - GDGEventsQueueHandler :907380469
2023-06-09 10:27:28 default        INFO  - c.n.b.g.r.s.ScriptControlTower          - Added jobExecutor [a65c2791-864f-43c9-972a-b5f2353389e6] to Sync Script Control Tower.
2023-06-09 10:27:28 default        INFO  - c.n.b.g.r.j.s.JobExecutor              - a65c2791-864f-43c9-972a-b5f2353389e6 - worker :Thread-26 [1547512424]
2023-06-09 10:27:28 default        INFO  - c.n.b.g.r.j.s.JobExecutor              - Triggered script: INTCALC - [a65c2791-864f-43c9-972a-b5f2353389e6] - jobContext [419695287]
2023-06-09_10-27-29-613 | [JOB] INTCALC - Started
2023-06-09_10-27-29-651 | [STEP] STEP15 - Started
2023-06-09 10:27:29 default        ERROR - c.n.b.g.r.c.i.ExecutionControllerImpl   - Could not find program "CBACT04C" in the program registry.
2023-06-09 10:27:29 default        ERROR - c.n.b.g.r.c.i.ExecutionControllerImpl   - Could not find program "CBACT04C" in the program registry.
2023-06-09_10-27-29-760 | Program not found => not executed !
2023-06-09_10-27-29-761 | [STEP] STEP15 - Ended
2023-06-09_10-27-29-772 | [JOB] INTCALC - Ended
2023-06-09 10:27:29 default        INFO  - c.n.b.g.r.j.s.DefaultJobContext        - Job [419695287] - starting final operation
2023-06-09 10:27:29 default        INFO  - c.n.b.g.r.j.s.DefaultJobContext        - End of job [419695287]
2023-06-09 10:27:29 default        INFO  - c.n.b.g.r.s.ScriptControlTower         - Removed jobExecutor [a65c2791-864f-43c9-972a-b5f2353389e6] from Script Control Tower.
2023-06-09 10:27:29 default        INFO  - c.n.b.g.r.s.ScriptControlTower         - Remaining jobExecutors:0
```

> **Note**
>
> 同步呼叫應保留給短時間執行的工作。長時間運行的作業應該以異步方式啟動（請參閱下面的專用端點）。

以非同步方式啟動指令碼

- 支持的方法：GET/POST
- 路徑：`/triggerscript/{scriptId:.+}`
- 參數：
  - 要啟動之指令碼的識別碼
  - 可選：使用請求參數傳遞給腳本的參數（視為 aMap<String,String>）。給定的參數將被自動添加到調用的常規腳本的 https://docs.groovy-lang.org/latest/html /api/groovy/lang/Binding.html [綁定]。
- 與上述同步模式相反，端點不會等待作業執行完成以傳送回應。如果可以找到可用的執行緒，並立即將回應傳送給呼叫者，則會立即啟動作業執行項目，其中包含作業執行 ID (代表作業執行項目的唯一識別碼)，可用來查詢作業執行狀態或強制終止應該發生錯誤的工作執行項目。回應的格式為：

```
Triggered script <script identifier> [unique job execution id] @ <date and time>
```

- 由於作業非同步執行依賴固定數量有限的執行緒，因此如果找不到可用的執行緒，工作執行可能不會啟動。在這種情況下，返回的消息看起來像：

```
Script [<script identifier>] NOT triggered - Thread limit reached (<actual thread
 limit>) - Please retry later or increase thread limit.
```

請參閱下面的settriggerthreadlimit端點以了解如何增加執行緒限制。

回應範例：

```
Triggered script INTCALC [d43cbf46-4255-4ce2-aac2-79137573a8b4] @ 06-12-2023 16:26:15
```

唯一的作業執行識別碼允許在必要時快速擷取伺服器記錄檔中的相關記錄項目。下面詳述的其他幾個端點也使用它。

列出觸發的腳本

- 支持的方法：GET
- 路徑：/triggeredscripts/{status:.+}，/triggeredscripts/{status:.+}/{namefilter}
- 引數：
  - 狀態 (必要)：要擷取之觸發程序檔的狀態。可能的值是：
    - all：顯示所有工作執行詳細資訊，無論工作是否仍在執行中。
    - running：僅顯示目前正在執行的工作的工作詳細資訊。
    - done：僅顯示執行結束之工作的工作詳細資訊。
    - killed：僅顯示使用專用端點強制終止執行的作業的作業詳細資訊 (請參閱下文)。
    - triggered：僅顯示已觸發但尚未啟動之工作的工作詳細資訊。
    - failed：僅顯示其執行已標記為失敗的工作的工作詳細資訊。
    - _namefilter（可選）_：僅檢索給定腳本標識符的執行。
- 返回作業執行詳細信息的集合。JSON如需詳細資訊，請參閱Job 執行詳細訊息結構。

回應範例：

```
[
    {
      "scriptId": "INTCALC",
      "caller": "127.0.0.1",
```

```
      "identifier": "d43cbf46-4255-4ce2-aac2-79137573a8b4",
      "startTime": "06-12-2023 16:26:15",
      "endTime": "06-12-2023 16:26:15",
      "status": "DONE",
      "executionResult": "{ \"exitCode\": -1, \"stepName\": \"STEP15\", \"program\":
\"CBACT04C\", \"status\": \"Error\" }",
      "executionMode": "ASYNCHRONOUS"
    }
 ]
```

擷取工作執行細節

- 支持的方法：GET
- 路徑：`/getjobexecutioninfo/{jobexecutionid:.+}`
- 引數：
  - jobexecutionid (必要)：用來擷取對應工作執行詳細資訊的唯一工作執行識別碼。
- 傳回：代表單一工作執行詳細資訊的JSON字串 (請參閱Job 執行詳細訊息結構)，如果找不到指定識別碼的工作執行詳細資訊，則為空白回應。

列出可以殺死的異步啟動腳本

- 支持的方法：GET
- 路徑：`/killablescripts`
- 返回作業執行標識符的集合，這些作業已異步啟動，這些作業仍在運行並且可以被強制終止（請參閱下面的`/kill`端點）。

列出可以殺死的同步啟動腳本

- 支持的方法：GET
- 路徑：`/killablesyncscripts`
- 返回作業執行標識符的集合，這些作業已同步啟動，仍在運行並且可以強行終止（請參閱下面的`/kill`端點）。

殺死給定的工作執行

- 支持的方法：GET

- 路徑：`/kill/{identifier:.+}`
- arguments：作業執行 ID (必要)：唯一的作業執行識別碼，用來指向要強制終止的工作執行。
- 傳回：詳細說明工作執行終止嘗試結果的文字訊息；訊息將包含指令碼識別碼、工作執行唯一識別碼，以及執行終止發生的日期和時間。如果找不到給定標識符的正在運行的作業執行，則會返回錯誤消息。

> ⚠ Warning
>
> - 運行時盡最大努力殺死目標作業執行很好。因此，來自 /kill 端點的響應可能需要一些時間才能到達調用者，因為 B AWS lu Age 運行時將嘗試最大程度地減少殺死工作的業務影響。
> - 強制終止工作執行不應輕心完成，因為它可能會產生直接的業務後果，包括可能的數據丟失或損壞。它應該保留給定的作業執行已經橫向和數據修復手段清楚識別的情況下。
> - 殺死工作應該導致進一步的調查（驗屍後分析），以找出出錯的原因並採取適當的補救措施。
> - 在任何情況下，嘗試終止正在運行的作業都將記錄在服務器日誌中，並帶有警告級別消息。

列出現有的檢查點以進行重新啟動

Job 重新啟動性依賴於指令碼在中註冊檢查點的能力，`CheckpointRegistry`以追蹤工作執行進度。如果作業執行無法正確結束，並且已註冊重新啟動檢查點，則只需從最後一個已知的已註冊檢查點重新啟動作業執行 (而不必執行檢查點上方的步驟)。

- 支持的方法：GET
- 路徑：`/restarts`
- 返回現有的重新啟動點列表，該列表可用於重新啟動其執行未正確結束的作業，作為 html 頁面。如果沒有任何腳本註冊檢查點，則頁面內容將是「沒有註冊的檢查點」。

重新啟動工作 (同步)

- 支持的方法：GET
- 路徑：`/restart/{hashcode}`
- 參數：哈希碼（整數-強制性）：重新啟動先前中止的作業執行，使用提供的哈希碼作為檢查點值（請參閱上面的`/restarts`端點以了解如何檢索有效的檢查點值）。

- 返回：請參閱上面的script返回說明。

重新啟動工作 (非同步)

- 支持的方法：GET
- 路徑：/triggerrestart/{hashcode}
- 參數：哈希碼（整數-強制性）：重新啟動先前中止的作業執行，使用提供的哈希碼作為檢查點值（請參閱上面的/restarts端點以了解如何檢索有效的檢查點值）。
- 返回：請參閱上面的triggerscript返回說明。

設定非同步工作執行的執行緒限制

工作非同步執行依賴於中的專用執行緒集區JVM。該集區對於可用執行緒數目有固定的限制。使用的具有根據主機功能（數量CPUs，可用內存等）來調整限制的能力。默認情況下，線程限制設置為 5 個線程。

- 支持的方法：GET
- 路徑：/settriggerthreadlimit/{threadlimit:.+}
- 引數 (整數)：要套用的新執行緒限制。必須是嚴格的正整數。
- 返回一個 message (String) 給出新的線程限制和前一個，或者如果提供的線程限制值無效（不是嚴格正整數），則返回 en 錯誤消息。

回應範例：

```
Set thread limit for Script Tower Control to 10 (previous value was 5)
```

計算目前執行的觸發工作執行項目

- 支持的方法：GET
- 路徑：/countrunningtriggeredscripts
- 傳回訊息，指出以非同步方式啟動的執行中作業數目和執行緒限制 (也就是可同時執行的觸發工作數目上限)。

回應範例：

```
0 triggered script(s) running (limit =10)
```

> **Note**
>
> 這可用於在啟動作業之前檢查是否尚未達到線程限制（這將阻止作業啟動）。

清除工作執行項目資訊

只要伺服器啟動，工作執行資訊就會保留在伺服器記憶體中。從內存中清除最舊的信息可能很方便，因為它們不再相關; 這是此端點的目的。

- 支持的方法：GET
- 路徑：`/purgejobinformation/{age:.+}`
- 引數：嚴格正整數值，代表要清除之資訊的年齡 (以小時為單位)。
- 傳回包含下列資訊的訊息：
  - 儲存已清除工作執行資訊以供存檔之清除檔案的名稱。
  - 已清除的工作執行資訊數目。
  - 備忘錄中剩餘作業執行資訊的數量

度量端點

JVM

此端點會傳回與JVM.

- 支持的方法：GET
- 路徑：`/metrics/jvm`
- 引數：無
- 傳回包含下列資訊的訊息：
  - threadActiveCount: 作用中執行緒的數目。
  - jvmMemoryUsed：Java 虛擬機器使用中的記憶體。
  - jvmMemoryMax：Java 虛擬機器允許的最大記憶體。
  - jvmMemoryFree：Java 虛擬機器目前未使用的可用記憶體。

Session (工作階段)

此端點傳回與目前開啟的HTTP工作階段相關的指標。

- 支持的方法：GET
- 路徑：/metrics/session
- 引數：無
- 傳回包含下列資訊的訊息：
  - sessionCount：伺服器目前維護的作用中使用者工作階段數目。

批次

- 支持的方法：GET
- 路徑：/metrics/batch
- 引數：
  - startTimestamp （可選，數字）：用於資料篩選的起始時間戳記。
  - endTimestamp （可選，數字）：資料篩選的結束時間戳記。
  - 頁面（可選，數量）：頁碼分頁。
  - pageSize （可選，數量）：在分頁每頁的項目數。
- 傳回包含下列資訊的訊息：
  - 內容：批次執行測量結果清單。
  - pageNumber：分頁中的當前頁碼。
  - pageSize：每頁顯示的項目數。
  - totalPages: 可用的總頁數。
  - numberOfElements：目前頁面上的項目計數。
  - last：布爾標誌的最後一頁。
  - 第一：布爾標誌的第一頁。

交易

- 支持的方法：GET
- 路徑：/metrics/transaction
- 引數：

  - startTimestamp （可選，數字）：用於資料篩選的起始時間戳記。
  - endTimestamp （可選，數字）：資料篩選的結束時間戳記。
  - 頁面（可選，數量）：頁碼分頁。
  - pageSize （可選，數量）：在分頁每頁的項目數。
- 傳回包含下列資訊的訊息：
  - content：交易執行測量結果清單。
  - pageNumber：分頁中的當前頁碼。
  - pageSize：每頁顯示的項目數。
  - totalPages: 可用的總頁數。
  - numberOfElements：目前頁面上的項目計數。
  - last：布爾標誌的最後一頁。
  - 第一：布爾標誌的第一頁。

其他端點

使用這些端點列出已註冊的程式或服務、探索健全狀態，以及管理JICS交易。

主題

- 列出已註冊程式
- 上市註冊服務
- 運作狀態
- 列出可用的JICS交易
- 啟動交JICS易
- 啟動交JICS易 (替代)
- 列出作用中階段

列出已註冊程式

- 支持的方法：GET
- 路徑：`/programs`
- 返回已註冊程序的列表，作為一個 html 頁面。每個程序由其主程序標識符指定。現代化的舊程序和實用程序（IDCAMSIEBGENER等）都在列表中返回。請注意，可用的實用程序將取決於您的

tomcat 服務器上部署的實用程序 Web 應用程序。例如，z/OS 公用程式支援方案可能不適用於現代化 iSeries 資產，因為它們不相關。

上市註冊服務

- 支持的方法：GET
- 路徑：`/services`
- 以 html 頁面的形式傳回已註冊執行階段服務的清單。給定的服務是由 AWS Blu Age 運行時作為實用程序提供的，可以在 groovy 腳本中使用。Blusam 加載服務（從舊數據集創建 Blusam 數據集）屬於該類別。

回應範例：

```
<p>BluesamESDSFileLoader</p><p>BluesamKSDSFileLoader</p><p>BluesamRRDSFileLoader</p>
```

運作狀態

- 支持的方法：GET
- 路徑：`/`
- 返回一個簡單的消息，表明 gapwalk 應用程序已啟動並正在運行（ ）`Jics application is running.`

列出可用的JICS交易

- 支持的方法：GET
- 路徑：`/transactions`
- 返回一個 HTML 頁面，列出所有可用的JICS交易。這只適用於具有JICS元素的環境（舊式元CICS素的現代化）。

回應範例：

```
<p>INQ1</p><p>MENU</p><p>MNT2</p><p>ORD1</p><p>PRNT</p>
```

啟動交JICS易

- 支持的方法：GET，POST

- 路徑：`/jicstransrunner/{jtrans:.+}`
- 參數：
  - JICS事務標識符（字符串，必需）：要啟動的JICS交易的標識符（最多 8 個字符）
  - 需要：額外的輸入數據傳遞給交易，作為一個地圖<String, Object>。此地圖的內容將用於饋送JICS交易將使用的內容。COMMAREA如果不需要任何資料來執行交易，則對應可以是空的。
  - 選用性：Http 標頭項目，用於自訂指定交易的執行環境。支援下列標頭鍵：
    - `jics-channel`：此交易啟動將啟動之程式所使用的名稱。JICS CHANNEL
    - `jics-container`：要用於此JICS交易啟動的名稱。JICS CONTAINER
    - `jics-startcode`: 要在JICS交易開始時使用的 STARTCODE (字串，最多 2 個字元)。請參STARTCODE閱以取得可能的值 (向下瀏覽頁面)。
    - `jicxa-xid`：由呼叫者起始的「全域交易」(XA) 的 (X /Open 交易識別碼XID結構)，目前JICS交易啟動將參與其中。XID
- 返回：`com.netfective.bluage.gapwalk.rt.shared.web.TransactionResultBean`JSON序列化，代表JICS交易啟動的結果。

如需結構詳細資訊的詳細資訊，請參閱異動啟動結果結構。

啟動交JICS易 (替代)

- 支持的方法：GET，POST
- 路徑：`/jicstransaction/{jtrans:.+}`
- 參數：

  JICS交易識別碼 (字串，必要)

  要啟動之JICS交易的識別碼 (最多 8 個字元)

  需要：額外的輸入數據傳遞給交易，作為地圖<String, Object>

  此地圖的內容將用於饋送JICS交易將使用的內容。COMMAREA如果不需要任何資料來執行交易，則對應可以是空的。

  選用性：Http 標頭項目，用於自訂指定交易的執行環境。

  支援下列標頭鍵：
  - `jics-channel`：此交易啟動將啟動之程式所使用的名稱。JICS CHANNEL
  - `jics-container`：要用於此JICS交易啟動的名稱。JICS CONTAINER

  - `jics-startcode`: 要在JICS交易開始時使用的 STARTCODE (字串，最多 2 個字元)。如需可能的值，請參閱 STARTCODE(向下瀏覽頁面)。
  - `jicxa-xid`：由呼叫者起始的「全域交易」(XA) 的 (X /Open 交易識別碼XID結構)，目前JICS交易啟動將參與其中。XID
- 返回：`com.netfective.bluage.gapwalk.rt.shared.web.RecordHolderBean`JSON序列化，代表JICS事務啟動的結果。結構的詳細資訊可在中找到異動啟動記錄結果結構。

列出作用中階段

- 支持的方法：GET，POST
- 路徑：`/activesessionlist`
- 參數：無
- return：JSON序列化`com.netfective.bluage.gapwalk.application.web.sessiontracker.SessionTrackerOb`列表，代表活動用戶會話的列表。當會話跟踪被禁用，一個空列表將被返回。

Job 佇列相關端點

Job 佇列是 AS4 00 個工作提交機制的 AWS 藍光時代支援。Job 佇列用於 AS4 00 中，在特定執行緒集區上執行作業。工作佇列是由名稱和最大執行緒數目所定義，這些執行緒數目對應於該佇列上可同時執行的程式數目上限。如果佇列上送出的工作超過最大執行緒數目，工作將會等待執行緒可用。

如需佇列中工作狀態的詳盡清單，請參閱佇列中工作的可能狀態。

工作佇列上的作業會透過下列專用端點來處理。您可以使用以下根目錄從 Gapwalk 應URL用程式呼叫這些作業URL：`http://server:port/gapwalk-application/jobqueue`

主題

- 列出可用的佇列
- 啟動或重新啟動工作佇列
- 提交工作以進行啟動
- 列出所有提交的工作
- 釋放所有「保留」的工作
- 釋放指定工作名稱的所有「保留」工作
- 釋出工作編號的指定工作

- 依重複排程送出工作
- 列出所有提交的重複工作
- 取消重複工作的排程

列出可用的佇列

- 支持的方法：GET
- 路徑：`list-queues`
- 返回可用隊列及其狀態的列表，作為鍵值的JSON列表。

回應範例：

```
{"Default":"STAND_BY","queue1":"STARTED","queue2":"STARTED"}
```

工作佇列的可能狀態為：

STAND_ 通過

工作佇列正在等待啟動。

STARTED

工作佇列已啟動並正在執行中。

UNKNOWN

無法判斷工作佇列狀態。

啟動或重新啟動工作佇列

- 支持的方法：POST
- 路徑：`/restart/{name}`
- 參數：隊列的名稱被啟動/重新啟動，作為一個字符串-強制性。
- 端點不會返回任何內容，而是依賴 http 狀態來指示啟動/重新啟動操作的結果：

  HTTP200

  啟動/重新啟動操作順利：給定的作業隊列現在STARTED是。

HTTP404

工作佇列不存在。

HTTP503

在開始/重新啟動嘗試期間發生異常（應檢查服務器日誌以找出錯誤）。

提交工作以進行啟動

- 支持的方法：POST
- 路徑：`/submit`
- 參數：強制性作為請求主體，一個`com.netfective.bluage.gapwalk.rt.jobqueue.SubmitJobMessage`對象的JSON序列化。如需詳細資訊，請參閱提交工作和排程工作輸入。
- 返回：一個JSON包含原始`SubmitJobMessage`和一個日誌，指示作業是否已提交或沒有。

列出所有提交的工作

- 支持的方法：GET
- 路徑：`/list-jobs?status={status}&size={size}&page={page}&sort={sort}`
- 引數：
  - 頁面：要擷取的頁碼 (預設值 = 1)
  - 大小：頁面大小（默認 = 50，最大 = 300）
  - 排序：「工作」的順序。 (預設值 =「executionId」)。「executionId」是目前唯一支援的值
  - status:（可選）如果存在，它將過濾狀態。
- 返回：所有計劃作業的列表，作為一個JSON字符串。如需回應範例，請參閱已排程的工作回應清單。

釋放所有「保留」的工作

- 支持的方法：POST
- 路徑：`/release-all`
- 傳回：指出發行嘗試作業結果的訊息。這裡有兩種可能的情況：
  - HTTP200 和一條消息「所有工作都成功發布了！」 如果所有工作都已成功釋放。

- HTTP503 和一條消息「工作未釋放。發生未知的錯誤。有關更多詳細信息，請參閱日誌「如果發布嘗試出現問題。

釋放指定工作名稱的所有「保留」工作

對於給定的作業名稱，可以提交多個作業，具有不同的作業編號（作業運行的單一性由一對夫婦授予<job name, job number>）。端點將嘗試釋放具有指定工作名稱（「保留」）的所有工作提交。

- 支持的方法：POST
- 路徑：`/release/{name}`
- 引數：要尋找的工作名稱，以字串形式。強制性。
- 傳回：指出發行嘗試作業結果的訊息。這裡有兩種可能的情況：
  - HTTP200 和一條消息「組中的工作<name>（<number of released jobs>）成功發布！」 工作已成功釋放。
  - HTTP503 和一條消息「組中的作業<name>未釋放。發生未知的錯誤。有關更多詳細信息，請參閱日誌「如果發布嘗試出現問題。

釋出工作編號的指定工作

端點將嘗試釋放給定夫婦的唯一作業提交「處於保留狀態」<job name, job number>。

- 支持的方法：POST
- 路徑：`/release/{name}/{number}`
- 引數：

  name

  要尋找的工作名稱，以字串形式顯示。強制性。

  number

  要尋找的工作編號，以整數表示。強制性。

  傳回

  一則訊息，指出發行嘗試作業的結果。這裡有兩種可能的情況：
  - HTTP200 和一條消息「」Job <name/number> 成功發布！」 如果工作已成功釋放。
  - HTTP<name/number>503 和一條消息「Job > 未發布。發生未知的錯誤。有關更多詳細信息，請參閱日誌「如果發布嘗試出現問題。

依重複排程送出工作

排定將使用重複排程執行的工作。

- 支持的方法：POST
- 路徑：/schedule
- 參數：請求主體必須包含com.netfective.bluage.gapwalk.rt.jobqueue.SubmitJobMessage對象的JSON序列化。

列出所有提交的重複工作

- 支持的方法：GET
- 路徑：/schedule/list?status={status}&size={size}&page={page}&sort={sort}
- 引數：
  1. 頁面：要擷取的頁碼 (預設值 = 1)
  2. 大小：頁面大小（默認 = 50，最大 = 300）
  3. 排序：「工作」的順序。（默認 =「ID」）。「id」是目前唯一支持的值。
  4. status:（可選）如果存在，它將過濾狀態。可能的值是第 1 部分中提到的值。
  5. status:（可選）如果存在，它將過濾狀態。可能的值是第 1 部分中提到的值。
  6. 返回：所有計劃作業的列表，作為一個JSON字符串。

取消重複工作的排程

移除在重複排程上建立的工作。工作排程狀態設定為INACTIVE。

- 支持的方法：GET
- 路徑：/schedule/remove/{schedule_id}
- 引數：schedule_id，要移除的已排程工作的識別碼。

Blusam 應用程式主控台端點 REST

在本節中，您可以了解 Blusam 應用程序控制台，該控制台API旨在簡化現代VSAM化數據集的管理。Blusam Web 應用程式的端點會使用根路徑。/bac

主題

- 資料集相關端點
- 大量資料集相關端點
- 記錄
- 遮罩
- 其他
- BAC使用者管理端點

資料集相關端點

使用下列端點建立或管理特定資料集。

主題

- 建立資料集
- 上傳檔案
- 載入資料集 (POST)
- 載入資料集 (GET)
- 從 Amazon S3 儲存貯體載入資料集
- 將資料集匯出到 Amazon S3 儲存貯體
- 清除資料集
- 刪除資料集
- 計數資料集記錄

建立資料集

您可以使用此端點建立資料集定義。

- 支持的方法：POST
- 需要驗證和 ROLE _ ADMIN 角色。
- 路徑：`/api/services/rest/bluesamservice/createDataSet`
- 引數：

  name

  （必要，字串）：資料集的名稱。

type

（必需，字符串）：數據集類型。可能的值為：ESDS、KSDS、RRDS。

recordSize

（可選，字串）：資料集中每個記錄的大小上限。

fixedLength

（可選，布林值）：指出記錄長度是否固定。

compression

(選擇性，布林值)：指出資料集是否已壓縮。

cacheEnable

(選擇性，布林值)：指出資料集是否啟用快取功能。

alternativeKeys

（可選，鍵列表）：

- 偏移量（必需，數字）
- 長度（必填，數字）
- 姓名（必填，號碼）

- 返回表示新創建的數據集的JSON文件。

樣品請求：

```
POST /api/services/rest/bluesamservice/createDataSet
{
  "name": "DATASET",
  "checked": false,
  "records": [],
  "primaryKey": {
    "name": "PK"
  },
  "alternativeKeys": [
    {
      "offset": 10,
      "length": 10,
      "name": "ALTK_0"
    }
```

```
  ],
  "type": "ESDS",
  "recordSize": 10,
  "compression": true,
  "cacheEnable": true
}
```

回應範例：

```
{
    "dataSet": {
      "name": "DATASET",
      "checked": false,
      "nbRecords": 0,
      "keyLength": -1,
      "recordSize": 10,
      "compression": false,
      "fixLength": true,
      "type": "ESDS",
      "cacheEnable": false,
      "cacheWarmup": false,
      "cacheEviction": "100ms",
      "creationDate": 1686744961234,
      "modificationDate": 1686744961234,
      "records": [],
      "primaryKey": {
        "name": "PK",
        "offset": null,
        "length": null,
        "columns": null,
        "unique": true
      },
      "alternativeKeys": [
        {
          "offset": 10,
          "length": 10,
          "name": "ALTK_0"
        }
      ],
      "readLimit": 0,
      "readEncoding": null,
      "initCharacter": null,
      "defaultCharacter": null,
```

```
      "blankCharacter": null,
      "strictZoned": null,
      "decimalSeparator": null,
      "currencySign": null,
      "pictureCurrencySign": null
    },
    "message": null,
    "result": true
  }
```

上傳檔案

您可以使用此端點將檔案上傳到伺服器。該文件存儲在對應於每個特定用戶的臨時文件夾中。每次需要上傳檔案時，請使用此端點。

- 支持的方法：POST
- 需要驗證和 ROLE _ ADMIN 角色。
- 路徑：`/api/services/rest/bluesamservice/upload`
- 引數：

  file

  （必要，多部分/表單數據）：要上傳的文件。
- 返回反映上傳狀態的布爾值

載入資料集 (POST)

使用建立`createDataSet`資料集定義之後，您可以將與上載檔案相關聯的記錄載入特定的資料集。

- 支持的方法：POST
- 需要驗證和 ROLE _ ADMIN 角色。
- 路徑：`/api/services/rest/bluesamservice/loadDataSet`
- 引數：

  name

  （必要，字串）：資料集的名稱。
- 返回請求和加載的數據集的狀態。

載入資料集 (GET)

- 支持的方法：GET
- 需要驗證和 ROLE _ ADMIN 角色。
- 路徑：/api/services/rest/bluesamservice/loadDataSet
- 引數：

  name

  （必要，字串）：資料集的名稱。

  資料集檔案

  （必要，字串）：資料集檔案名稱。
- 返回請求和加載的數據集的狀態。

從 Amazon S3 儲存貯體載入資料集

使用 Amazon S3 儲存貯體中的清單貓檔案載入資料集。

- 支持的方法：GET
- 需要驗證和 ROLE _ ADMIN 角色。
- 路徑：/api/services/rest/bluesamservice/loadDataSetFromS3
- 引數：

  listcatFileS3 地點

  （必要，字串）：清單貓檔案的 Amazon S3 位置。

  datasetFileS3 地點

  (必要，字串)：資料集檔案的 Amazon S3 位置。

  region

  （必要，字符串）：存 AWS 區域 放文件的 Amazon S3。
- 返回新創建的數據集

樣品請求：

```
/BAC/api/services/rest/bluesamservice/loadDataSetFromS3?region=us-
east-1&listcatFileS3Location=s3://bucket-name/listcat.json&datasetFileS3Location=s3://
bucket-name/dataset.DAT
```

將資料集匯出到 Amazon S3 儲存貯體

將資料集匯出到指定的 Amazon S3 儲存貯體。

- 支持的方法：GET
- 需要驗證和 ROLE _ ADMIN 角色。
- 路徑：/api/services/rest/bluesamservice/exportDataSetToS3
- 引數：

  s3Location

  (必要，字串)：要將資料集匯出到的 Amazon S3 位置。

  datasetName

  (必要，字串)：要匯出的資料集名稱。

  region

  （必需，字符串）：Amazon S3 存儲桶 AWS 區域 的。
- 返回導出的數據集

樣品請求：

```
/BAC/api/services/rest/bluesamservice/exportDataSetToS3?region=eu-
west-1&s3Location=s3://bucket-name/dump&datasetName=dataset
```

清除資料集

清除資料集中的所有記錄。

- 支持的方法：POSTGET
- 需要驗證和 ROLE _ ADMIN 角色。
- 路徑：/api/services/rest/bluesamservice/clearDataSet
- 引數：

name

（必要，字串）：要清除的資料集名稱。

- 返回請求的狀態。

刪除資料集

刪除資料集定義和記錄。

- 支持的方法：POST
- 需要驗證和 ROLE _ ADMIN 角色。
- 路徑：`/api/services/rest/bluesamservice/deleteDataSet`
- 引數：

  name

  （必要，字串）：要刪除的資料集的名稱。
- 返回請求和刪除的數據集的狀態。

計數資料集記錄

此端點返回與數據集關聯的記錄數。

- 支持的方法：POST
- 需要驗證和 ROLE _ USER 角色。
- 路徑：`/api/services/rest/bluesamservice/countRecords`
- 引數：

  name

  （必要，字串）：資料集的名稱。
- 返回：記錄數

大量資料集相關端點

使用下列端點一次建立或管理多個資料集。

主題

- 匯出資料集 (GET)
- 匯出資料集 (POST)
- 建立多個資料集
- 列出所有資料集
- 直接列出所有資料集
- 依頁面直接列出所有資料集
- 串流資料集
- 刪除所有資料集
- 從列表貓文件獲取數據集定義
- 從上傳的列表 cat 文件中獲取數據集定義
- 取得資料集
- 從JSON文件加載列表貓

匯出資料集 (GET)

- 支持的方法：GET
- 需要驗證和 ROLE _ USER 角色。
- 路徑：`/api/services/rest/bluesamservice/exportDataSet`
- 引數：

  datasetName

  (必要，字串)：要匯出的資料集名稱。

  datasetOutputFile

  （必要，字串）：要在伺服器上儲存匯出資料集的資料夾路徑。

  rdw

  (必要，布林值)：是否要將記錄描述元字 (RDW) 成為匯出記錄的一部分。如果資料集具有固定長度的記錄，則會忽略此參數的值。
- 傳回要求的狀態，以及包含匯出資料集 (如果有的話) 的檔案路徑。如果資料集在回應中為 null，表示系統無法找到具有指定名稱的資料集。

匯出資料集 (POST)

- 支持的方法：POST
- 需要驗證和 ROLE _ USER 角色。
- 路徑：/api/services/rest/bluesamservice/exportDataSet
- 引數：

  dumpParameters

  （必填BACReadParameters）：藍光讀取參數。
- 傳回匯出資料集的狀態。

建立多個資料集

- 支持的方法：POST
- 需要驗證和 ROLE _ ADMIN 角色。
- 路徑：/api/services/rest/bluesamservice/createAllDataSets
- 引數：
  - 資料集清單

    name

    （必要，字串）：資料集的名稱。

    type

    （必需，字符串）：數據集類型。可能的值為：ESDS、KSDS、RRDS。

    recordSize

    （可選，字串）：資料集中每個記錄的大小上限。

    fixedLength

    （可選，布林值）：指出記錄長度是否固定。

    compression

    (選擇性，布林值)：指出資料集是否已壓縮。

    cacheEnable

    (選擇性，布林值)：指出資料集是否啟用快取功能。

- 返回：請求的狀態和新創建的數據集。

列出所有資料集

- 支持的方法：GET
- 需要驗證和 ROLE _ USER 角色。
- 路徑：/api/services/rest/bluesamservice/listDataSet
- 引數：無
- 返回：請求的狀態和數據集的列表。

直接列出所有資料集

- 支持的方法：GET
- 需要驗證和 ROLE _ USER 角色。
- 路徑：/api/services/rest/bluesamservice/directListDataSet
- 引數：無
- 返回：請求的狀態和數據集的列表。

依頁面直接列出所有資料集

- 支持的方法：GET
- 需要驗證和 ROLE _ USER 角色。
- 路徑：/api/services/rest/bluesamservice/directListDataSetByPage
- 引數：

  datasetName

  （必要，字串）：資料集的名稱。

  pageNumber

  （必需，int）：頁碼。

  pageSize

  （必需，int）：頁面大小。
- 返回：請求的狀態和數據集的列表。

串流資料集

- 支持的方法：GET
- 需要驗證和 ROLE _ ADMIN 角色。
- 路徑：/api/services/rest/bluesamservice/streamDataset
- 引數：

  datasetName

  （必要，字串）：資料集的名稱。
- 返回：請求的數據集的流。

刪除所有資料集

- 支持的方法：POST
- 需要驗證和 ROLE _ ADMIN 角色。
- 路徑：/api/services/rest/bluesamservice/removeAll
- 引數：無
- 返回：布爾值，表示請求的狀態。

從列表貓文件獲取數據集定義

- 支持的方法：POST
- 需要驗證和 ROLE _ ADMIN 角色。
- 路徑：/api/services/rest/bluesamservice/getDataSetsDefinitionFromListcat
- 引數：

  paramFilePath

  （必要，字符串）：列表貓文件的路徑。
- 返回：數據集列表

從上傳的列表 cat 文件中獲取數據集定義

- 支持的方法：POST
- 需要驗證和 ROLE _ ADMIN 角色。

- 路徑：/api/services/rest/bluesamservice/getDataSetsDefinitionFromUploadedListcat
- 引數：無
- 返回：數據集列表

取得資料集

- 支持的方法：GET
- 需要驗證和 ROLE _ USER 角色。
- 路徑：/api/services/rest/bluesamservice/getDataSet
- 引數：

  name

  （必要，字串）：資料集的名稱。
- 返回請求的數據集。

從JSON文件加載列表貓

- 支持的方法：GET
- 需要驗證和 ROLE _ ADMIN 角色。
- 路徑：/api/services/rest/bluesamservice/loadListcatFromJsonFile
- 引數：

  filePath

  （必要，字符串）：列表貓文件的路徑。
- 返回：數據集列表

記錄

使用下列端點建立或管理資料集內的記錄。

主題

- 建立記錄
- 讀取資料集

- 刪除記錄
- 更新記錄
- 儲存記錄
- 驗證記錄
- 獲取記錄樹

建立記錄

您可以使用此端點來建立新記錄。

- 支持的方法：POST
- 需要驗證和 ROLE _ USER 角色。
- 路徑：`/api/services/rest/crud/createRecord`
- 引數：

  dataset

  (必要， DataSet)：資料集物件

  遮罩

  （必填，遮罩）：遮色片物件。
- 返回請求和創建的記錄的狀態。

讀取資料集

您可以使用此端點讀取資料集。

- 支持的方法：POST
- 需要驗證和 ROLE _ USER 角色。
- 路徑：`/api/services/rest/crud/readDataSet`
- 引數：

  dataset

  (必要， DataSet)：資料集物件。
- 返回請求的狀態，並與記錄的數據集。

刪除記錄

您可以使用此端點從資料集刪除記錄。

- 支持的方法：POST
- 需要驗證和 ROLE _ USER 角色。
- 路徑：`/api/services/rest/crud/deleteRecord`
- 引數：

  dataset

  (必要， DataSet)：資料集物件

  record

  (必要，記錄)：要刪除的記錄
- 傳回刪除的狀態。

更新記錄

您可以使用此端點更新與資料集關聯的記錄。

- 支持的方法：POST
- 需要驗證和 ROLE _ USER 角色。
- 路徑：`/api/services/rest/crud/updateRecord`
- 引數：

  dataset

  (必要， DataSet)：資料集物件

  record

  (必要，記錄)：要更新的記錄
- 返回請求的狀態，並與記錄的數據集。

儲存記錄

您可以使用此端點將記錄儲存至資料集並使用遮罩。

- 支持的方法：POST

- 需要驗證和 ROLE _ USER 角色。
- 路徑：/api/services/rest/crud/saveRecord
- 引數：

  dataset

  (必要， DataSet)：資料集物件

  record

  (必要，記錄)：要儲存的記錄
- 返回請求的狀態，並與記錄的數據集。

驗證記錄

使用此端點來驗證記錄。

- 支持的方法：POST
- 需要驗證和 ROLE _ USER 角色。
- 路徑：/api/services/rest/crud/validateRecord
- 引數：

  dataset

  (必要， DataSet)：資料集物件
- 返回請求的狀態，並與記錄的數據集。

獲取記錄樹

使用此端點取得記錄的階層式樹狀結構。

- 支持的方法：POST
- 需要驗證和 ROLE _ USER 角色。
- 路徑：/api/services/rest/crud/getRecordTree
- 引數：

  dataset

  (必要， DataSet)：資料集物件

record

（必需，記錄）：要獲取的記錄

- 返回請求的狀態和請求的記錄的分層樹。

遮罩

使用下列端點載入或套用遮罩至資料集。

主題

- 載入遮罩
- 套用遮罩
- 套用遮罩濾鏡

載入遮罩

您可以使用此端點擷取與特定資料集相關聯的所有遮罩。

- 支持的方法：POST
- 需要驗證和 ROLE _ USER 角色。
- 路徑：`/api/services/rest/crud/loadMasks`
- 路徑變數：

  recordSize: .../loadMasks/{recordSize}

  （可選，數字）：記錄大小，過濾器加載的掩碼符合此記錄大小
- 引數：

  dataset

  (必要， DataSet)：資料集物件
- 返回請求的狀態和掩碼的列表。

套用遮罩

您可以使用此端點將遮罩套用至特定資料集。

- 支持的方法：POST

- 需要驗證和 ROLE _ USER 角色。
- 路徑：/api/services/rest/crud/applyMask
- 引數：

  dataset

  (必要， DataSet)：資料集物件

  遮罩

  (必要，遮罩)：資料集物件
- 返回請求的狀態，並與應用的掩碼的數據集。

套用遮罩濾鏡

您可以使用此端點將遮罩和篩選器套用至特定資料集。

- 支持的方法：POST
- 需要驗證和 ROLE _ USER 角色。
- 路徑：/api/services/rest/crud/applyMaskFilter
- 引數：

  dataset

  (必要， DataSet)：資料集物件

  遮罩

  (必要，遮罩)：資料集物件
- 返回請求的狀態，並與應用的掩碼和過濾器的數據集。

其他

使用下列端點管理資料集的快取或檢查資料集特性

主題

- 檢查暖機快取
- 檢查緩存已啟用
- 啟用快取

- 檢查分配的RAM緩存
- 檢查持久性
- 檢查支援的資料集類型
- 檢查伺服器健康

檢查暖機快取

檢查特定資料集是否已啟用暖機快取。

- 支持的方法：POST
- 需要驗證和 ROLE _ ADMIN 角色。
- 路徑：`/api/services/rest/bluesamservice/warmupCache`
- 引數：

  name

  （必要，字串）：資料集的名稱。
- 返回：如果預熱緩存被啟用，否則返回 false。

檢查緩存已啟用

檢查是否已針對特定資料集啟用快取。

- 支持的方法：GET
- 需要驗證和 ROLE _ USER 角色。
- 路徑：`/api/services/rest/bluesamservice/isEnableCache`
- 引數：無
- 如果啟用了緩存，則返回 true。

啟用快取

- 支持的方法：GET
- 需要驗證以及 ROLE _ ADMIN 和 ROLE SUPER _ ADMIN 角色。
- 路徑：`/api/services/rest/bluesamservice/enableDisableCache/{enable}`
- 引數：

啟用

（必需，布爾值）：如果設置為 true，它將啟用緩存。

- 返回無

檢查分配的RAM緩存

您可以使用此端點擷取配置的RAM快取記憶體。

- 支持的方法：GET
- 需要驗證和 ROLE _ USER 角色。
- 路徑：/api/services/rest/bluesamservice/allocatedRamCache
- 引數：無
- 返回：將內存的大小作為字符串

檢查持久性

- 支持的方法：GET
- 需要驗證和 ROLE _ USER 角色。
- 路徑：/api/services/rest/bluesamservice/persistence
- 引數：無
- 返回：用作字符串的持久性

檢查支援的資料集類型

- 支持的方法：GET
- 路徑：/api/services/rest/bluesamservice/getDataSetTypes
- 需要驗證和 ROLE _ USER 角色。
- 引數：無
- 返回：支持的數據集類型列表作為字符串列表。

檢查伺服器健康

- 支持的方法：GET

- 路徑：/api/services/rest/bluesamserver/serverIsUp
- 引數：無
- 返回：無。HTTP響應狀態碼 200 表示服務器已啟動並運行。

BAC使用者管理端點

使用下列端點來管理使用者互動。

主題

- 登入使用者
- 驗證系統中是否至少存在一個用戶
- 記錄新使用者
- 取得使用者資訊
- 列出使用者
- 刪除使用者
- 將目前的使用者登出

登入使用者

- 支持的方法：POST
- 路徑：/api/services/security/servicelogin/login
- 引數：無
- 返回對com.netfective.bluage.bac.entities.SignOn象的JSON序列化，代表其憑據在當前請求中提供的用戶。密碼會從傳回物件的檢視中隱藏起來。指定給使用的角色正在列出。

回應範例：

```
{
     "login": "some-admin",
     "password": null,
     "roles": [
       {
         "id": 0,
         "roleName": "ROLE_ADMIN"
```

```
      }
    ]
  }
```

驗證系統中是否至少存在一個用戶

- 支持的方法：GET
- 路徑：/api/services/security/servicelogin/hasAccount
- 引數：無
- true如果至少已建立預設超級管理員使用者以外的其他使用者，則傳回布林值。false否則返回。

記錄新使用者

- 支持的方法：POST
- 需要驗證和 ROLE _ ADMIN 角色。
- 路徑：/api/services/security/servicelogin/recorduser
- 參數：代表要添加到存儲中的用戶的com.netfective.bluage.bac.entities.SignOn對象的JSON序列化。必須定義使用者的角色，否則使用者可能無法使用該BAC設施和端點。
- true如果成功創建用戶，則返回布爾值。false否則返回。
- 樣品請求JSON：

```
{
    "login": "simpleuser",
    "password": "simplepassword",
    "roles": [
      {
        "id": 2,
        "roleName": "ROLE_USER"
      }
    ]
  }
```

以下是兩個有效值roleName：

- ROLE_ADMIN: 可以管理 Blusam 的資源和使用者。
- ROLE_USER：可以管理 Blusam 資源，但不能管理使用者。

取得使用者資訊

- 支持的方法：GET
- 路徑：/api/services/security/servicelogin/userInfo
- 引數：無
- 返回當前連接的用戶的用戶名和角色

列出使用者

- 支持的方法：GET
- 需要驗證和 ROLE _ ADMIN 角色。
- 路徑：/api/services/security/servicelogin/listusers
- 引數：無
- 返回一個列表com.netfective.bluage.bac.entities.SignOn，序列化為JSON。

刪除使用者

> **Important**
>
> 這個操作無法復原。刪除的使用者將無法再次連線至BAC應用程式。

- 支持的方法：POST
- 需要驗證和 ROLE _ ADMIN 角色。
- 路徑：/api/services/security/servicelogin/deleteuser
- 參數：代表要從存儲中刪除的用戶的com.netfective.bluage.bac.entities.SignOn對象的JSON序列化。
- 返回布爾值，true如果用戶被成功刪除。

將目前的使用者登出

- 支持的方法：GET
- 路徑：/api/services/security/servicelogout/logout
- 引數：無

- {"success":true}如果目前使用者已成功登出，則傳回JSON訊息。相關的HTTP工作階段將會失效。

在 AWS 藍光時代管理JICS應用程序控制

該JICS組件是對舊CICS資源現代化的 AWS 藍光時代支持。JICS應用程式主控台 Web 應用程式專用於管理JICS資源。下列端點允許執行管理工作，而不必與JAC使用者介面互動。每當端點需要驗證時，要求都必須包含驗證詳細資料 (使用者名稱/密碼通常是根據基本驗證的要求)。應用JICS程式主控台 Web 應用程式的端點會使用根路徑/jac/。

主題

- JICS資源管理
- 其他
- JAC使用者管理端點

JICS資源管理

以下所有端點都與資JICS源管理相關，允許JICS管理員每天處理資源。

主題

- 列表JICSLISTS和 GROUPS
- 擷取JICS資源
- 清單 JICS GROUPS
- 給定JICSGROUPS的列表 LIST
- LISTJICS給定的資源 GROUP
- LISTJICS給定的資源GROUP（使用名稱的替代方法）
- 編輯擁有GROUPS的幾個 LISTS
- 刪除一個 LIST
- 刪除一個 GROUP
- 刪除一個 TRANSACTION
- 刪除一個 PROGRAM
- 刪除一個 FILE

- 刪除一個 TDQUEUE
- 刪除一個 TSMODEL
- 刪除圖元
- 創建一個 LIST
- 創建一個 GROUP
- 常見的RESOURCES建立考量
- 創建一個 TRANSACTION
- 創建一個 PROGRAM
- 創建一個 FILE
- 創建一個 TDQUEUE
- 創建一個 TSMODEL
- 建立元素
- 更新一個 LIST
- 更新一個 GROUP
- 常見的RESOURCES更新考量
- 更新一個 TRANSACTION
- 更新一個 PROGRAM
- 更新一個 FILE
- 更新一個 TDQUEUE
- 更新一個 TSMODEL
- 更新圖元
- Upsert 元素
- 擷取元素
- JICSCRUD操作

列表JICSLISTS和 GROUPS

LIST和GROUPS是JICS元件內的主要擁有容器資源。所有JICS資源都必須屬於GROUP. 群組可以屬於LISTS，但這不是強制性的。LISTS甚至可能不存在於給定的JICS環境中，但大多數情況下，LISTS都會為資源提供額外的組織層。如需有關資CICS源組織的詳細資訊，請參閱CICS資源。

- 支持的方法：GET
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：`/api/services/rest/jicsservice/listJicsListsAndGroups`
- 引數：無
- 返回：序列化 JicsContainer 對象的列表，包括LISTS和GROUPS，as JSON。

回應範例：

```
[
    {
      "name": "Resources",
      "children": [
        {
          "jacType": "JACList",
          "name": "MURACHS",
          "isActive": true,
          "children": [
            {
              "jacType": "JACGroup",
              "name": "MURACHS",
              "isActive": true,
              "children": []
            }
          ]
        },
        {
          "jacType": "JACGroup",
          "name": "TEST",
          "isActive": true,
          "children": []
        }
      ],
      "isExpanded": true
    }
  ]
```

擷取JICS資源

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER

- 路徑：/api/services/rest/jicsservice/retrieveJicsResources
- 引數：代表您要擷取的JICS資源的JSON裝載。這是一個com.netfective.bluage.jac.entities.request.RetrieveOperationRequest對象的JSON序列化。
- 傳回：序列化 JicsResource 物件的清單。物件會以沒有特定的順序傳回，而且具有不同類型PROGRAMTRANSACTION，例如FILE、、等等。

清單 JICS GROUPS

- 支持的方法：GET
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：/api/services/rest/jicsservice/listJicsGroups
- 引數：無
- 返回序列化 JicsContainer 對象（GROUPS）的列表。JSON在GROUPS沒有其擁有LIST資訊的情況下傳回。

回應範例：

```
[
    {
      "jacType": "JACGroup",
      "name": "MURACHS",
      "isActive": true,
      "children": []
    },
    {
      "jacType": "JACGroup",
      "name": "TEST",
      "isActive": true,
      "children": []
    }
  ]
```

給定JICSGROUPS的列表 LIST

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER

- 路徑：/api/services/rest/jicsservice/listGroupsForList
- 參數：有JSON效載荷，代表GROUPS您正在尋找JICSLIST的。這是一個com.netfective.bluage.jac.entities.JACList對象的JSON序列化。

  樣品請求：

```
{
    "jacType":"JACList",
    "name":"MURACHS",
    "isActive":true
  }
```

- 傳回附加至指LIST定之序列化 JicsContainer 物件 (GROUPS) 的清單。JSON在GROUPS沒有其擁有LIST資訊的情況下傳回。

  回應範例：

```
[
    {
      "jacType": "JACGroup",
      "name": "MURACHS",
      "isActive": true,
      "children": []
    }
  ]
```

LISTJICS給定的資源 GROUP

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：/api/services/rest/jicsservice/listResourcesForGroup
- 參數：有JSON效載荷，代表您正在尋找JICSGROUP的資源。這是一個com.netfective.bluage.jac.entities.JACGroup對象的JSON序列化。您不需要指定的所有欄位GROUP，但名稱是必要的。

  樣品請求：

```
{
    "jacType":"JACGroup",
```

```
    "name":"MURACHS",
    "isActive":true
  }
```

- 傳回序列化 JicsResource 物件的清單，由指定GROUP的擁有。物件會以沒有特定的順序傳回，而且具有不同類型 PROGRAMTRANSACTION，例如FILE、、等等。

LISTJICS給定的資源GROUP（使用名稱的替代方法）

- 支持的方法：POST
- 需要驗證
- 路徑：`/api/services/rest/jicsservice/listResourcesForGroupName`
- 參數：GROUP擁有您正在尋找的資源的名稱。
- 返回：序列化 JicsResource 對象的列表，由給定GROUP的擁有。物件會以沒有特定的順序傳回，而且具有不同的類型 PROGRAMTRANSACTION，例如FILE、、等等。

編輯擁有GROUPS的幾個 LISTS

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：`/api/services/rest/jicsservice/editGroupsList`
- 參數：LISTS與孩子的集合的JSON表示GROUPS;

  樣品請求：

```
[
  {
        "jacType": "JACList",
        "name": "MURACHS",
        "isActive": true,
        "children": [
          {
            "jacType": "JACGroup",
            "name": "MURACHS",
            "isActive": true,
            "children": []
          },
          {
            "jacType": "JACGroup",
```

```
            "name": "TEST",
            "isActive": true,
            "children": []
          }
        ]
  }
]
```

在編輯之前，只有名為 "MURACHS" 的群組屬於LIST命名的 "MURACHS"。透過此編輯，您可以將名為 "" 的群組「加入TEST」到LIST具名的 "MURACHS"。

- 返回一個布爾值。如果值為 'true'，則LISTS修改已成功保存在基礎JICS存儲中。

刪除一個 LIST

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：`/api/services/rest/jicsservice/deleteList`
- 引數：有JSON效載荷，代表JICSLIST要刪除。這是一個`com.netfective.bluage.jac.entities.JACList`對象的JSON序列化。
- 返回一個布爾值。如果值為「true」，則表示已在基礎JICS儲存裝置上成功執行LIST刪除作業。

刪除一個 GROUP

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：`/api/services/rest/jicsservice/deleteGroup`
- 引數：有JSON效載荷，代表JICSGROUP要刪除。這是一個`com.netfective.bluage.jac.entities.JACGroup`對象的JSON序列化。
- 返回一個布爾值。如果值為「true」，則表示已在基礎JICS儲存裝置上成功執行GROUP刪除作業。

刪除一個 TRANSACTION

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：`/api/services/rest/jicsservice/deleteTransaction`

- 引數：有JSON效載荷，代表要刪除的JICS交易。這是一個com.netfective.bluage.jac.entities.JACTransaction對象的JSON序列化。
- 返回一個布爾值。如果值為「true」，則表示已在基礎JICS儲存裝置上成功執行TRANSACTION刪除作業。

刪除一個 PROGRAM

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：/api/services/rest/jicsservice/deleteProgram
- 引數：有JSON效載荷，代表要刪除的JICS程式。這是一個com.netfective.bluage.jac.entities.JACProgram對象的JSON序列化。
- 返回一個布爾值。如果值為「true」，則表示已在基礎JICS儲存裝置上成功執行PROGRAM刪除作業。

刪除一個 FILE

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：/api/services/rest/jicsservice/deleteFile
- 參數：有JSON效載荷，代表要刪除的JICS文件。這是一個com.netfective.bluage.jac.entities.JACFile對象的JSON序列化。
- 返回一個布爾值。如果值為「true」，則表示已在基礎JICS儲存裝置上成功執行FILE刪除作業。

刪除一個 TDQUEUE

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：/api/services/rest/jicsservice/deleteTDQueue
- 引數：有JSON效載荷，代表JICSTDQUEUE要刪除。這是一個JSON序列化的網絡。藍色。JACTDQueue`物件。
- 返回一個布爾值。如果值為「true」，則表示已在基礎JICS儲存裝置上成功執行TDQUEUE刪除作業。

刪除一個 TSMODEL

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：`/api/services/rest/jicsservice/deleteTSModel`
- 引數：有JSON效載荷，代表JICSTSMODEL要刪除。這是一個JSON序列化的網絡。藍色。JACTSModel`物件。
- 返回一個布爾值。如果值為「true」，則表示已在基礎JICS儲存裝置上成功執行TSMODEL刪除作業。

刪除圖元

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：`/api/services/rest/jicsservice/deleteElements`
- 引數：代表要刪除之JICS元素的JSON裝載。
- 返回一個布爾值，其中`true`表示刪除已在基礎JICS存儲中成功操作。

創建一個 LIST

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：`/api/services/rest/jicsservice/createList`
- 引數：有JSON效載荷，代表JICSLIST要建立。這是一個JSON序列化的網絡。藍色。 JACList`物件。
- 返回一個布爾值。如果值為「true」，表示LIST已在基礎JICS儲存裝置中成功建立。

> **Note**
>
> LIST將永遠建立為空白。附加GROUPS到LIST將需要另一個操作。

創建一個 GROUP

- 支持的方法：POST

- 需要驗證和以下角色：ROLE_ADMIN，ROLE_ SUPER _ADMIN，ROLE_ USER
- 路徑：`/api/services/rest/jicsservice/createGroup`
- 引數：有JSON效載荷，代表JICSGROUP要建立。這是一個`com.netfective.bluage.jac.entities.JACGroup`對象的JSON序列化。
- 返回一個布爾值。如果值為「true」，表示GROUP已在基礎JICS儲存區中正確建立。

> **Note**
>
> GROUP將永遠建立為空白。附加RESOURCES至GROUP將需要額外的作業 (建立資源會自動將它們附加至指定的作業GROUP。

常見的RESOURCES建立考量

以下所有端點都與JICSRESOURCES創建和共享一些常見約束有關：在要發送到端點的請求有效負載中，必須對該`groupName`字段進行估值。

GROUP所有權限制：

如果不附加至現有群組，則無法建立任何資源，且端點會使用擷取此資源將附加至的群組。 groupName 必`groupName`須指向現有的名稱GROUP。如果未指向JICS基礎儲存區中的`groupName`現有群組，則會傳送含有 HTTP STATUS 400 的錯誤訊息。

一GROUP個內的單一性約束：

具有給定名稱的給定資源在給定組中必須是唯一的。每個資源建立端點將執行單一性檢查。如果給定的有效負載不遵守 unicity 約束，則端點將發送帶有 HTTP STATUS 400 (BADREQUEST) 的響應-請參閱下面的示例響應。

範例承載：您嘗試在 'ARIT' 群組中建立交易 'TEST'，但該群組中已存在具有該名稱的交易。

```
{
    "jacType":"JACTransaction",
    "name":"ARIT",
    "groupName":"TEST",
    "isActive":true
  }
```

您會收到下列錯誤回應：

```
{
    "timestamp": 1686759054510,
    "status": 400,
    "error": "Bad Request",
    "path": "/jac/api/services/rest/jicsservice/createTransaction"
  }
```

檢查伺服器記錄檔將確認問題的根源：

```
2023-06-14 18:10:54 default         TRACE - o.s.w.m.HandlerMethod
        - Arguments: [java.lang.IllegalArgumentException: Transaction already
 present in the group, org.springframework.security.web.header.HeaderWriterFilter
$HeaderWriterResponse@e34f6b8]
2023-06-14 18:10:54 default         ERROR - c.n.b.j.a.WebConfig                    -
 400
java.lang.IllegalArgumentException: Transaction already present in the group
 at
 com.netfective.bluage.jac.server.services.rest.impl.JicsServiceImpl.createElement(JicsServiceI
```

創建一個 TRANSACTION

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：`/api/services/rest/jicsservice/createTransaction`
- 引數：有JSON效載荷，代表JICSTRANSACTION要建立。這是一個`com.netfective.bluage.jac.entities.JACTransaction`對象的JSON序列化。
- 返回一個布爾值。如果值為「true」，表示TRANSACTION已在基礎JICS儲存裝置中成功建立。

創建一個 PROGRAM

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：`/api/services/rest/jicsservice/createProgram`
- 引數：有JSON效載荷，代表JICSPROGRAM要建立。這是一個`com.netfective.bluage.jac.entities.JACProgram`對象的JSON序列化。

- 返回一個布爾值。如果值為「true」，表示PROGRAM已在基礎JICS儲存裝置中成功建立。

創建一個 FILE

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：`/api/services/rest/jicsservice/createFile`
- 引數：有JSON效載荷，代表JICSFILE要建立。這是一個`com.netfective.bluage.jac.entities.JACFile`對象的JSON序列化。
- 返回一個布爾值。如果值為「true」，表示FILE已在基礎JICS儲存裝置中成功建立。

創建一個 TDQUEUE

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：`/api/services/rest/jicsservice/createTDQueue`
- 引數：有JSON效載荷，代表JICSTDQUEUE要建立。這是一個`com.netfective.bluage.jac.entities.JACTDQueue`對象的JSON序列化。
- 返回一個布爾值。如果值為「true」，表示TDQUEUE已在基礎JICS儲存裝置中成功建立。

創建一個 TSMODEL

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：`/api/services/rest/jicsservice/createTSModel`
- 引數：有JSON效載荷，代表JICSTSMODEL要建立。這是一個`com.netfective.bluage.jac.entities.JACTSModel`對象的JSON序列化。
- 返回一個布爾值，其中`true`表示元素的創建已在基礎JICS存儲中成功操作。

建立元素

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：`/api/services/rest/jicsservice/createElements`

- 引數：代表要建立之JICS元素的JSON裝載。
- 返回一個布爾值。如果值為 'true'，則在基礎JICS存儲中成功創建了元素。

更新一個 LIST

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：`/api/services/rest/jicsservice/updateList`
- 引數：有JSON效載荷，代表JICSLIST要更新。這是一個`com.netfective.bluage.jac.entities.JACList`對象的JSON序列化。沒有必要提供的孩子LIST; LIST 更新機制不會考慮孩子。
- 返回一個布爾值。如果值為「true」，表示LIST已在基礎JICS儲存區中成功更新。

更新 LIST 'isActive' 旗標會傳播到的所有擁有元素，也就是LIST，全部由這些元素GROUPS擁有，LIST且全部由這些GROUPS元素RESOURCES擁有。這是一種通過單次操作停用大量資源的便捷方法，可以通過多GROUPS個操作來停用。

更新一個 GROUP

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：`/api/services/rest/jicsservice/updateGroup`
- 引數：有JSON效載荷，代表JICSGROUP要更新。這是一個`com.netfective.bluage.jac.entities.JACGroup`對象的JSON序列化。沒有必要提供的子系GROUP，GROUP更新機制不會考慮到這一點。
- 返回一個布爾值。如果值為「true」，表示GROUP已在基礎JICS儲存區中成功更新。

> **Note**
>
> 更新 GROUP 'isActive' 旗標將會傳播至所有擁有的元素GROUP，也就是全部RESOURCESGROUP由. 這是通過給GROUP定的單個操作停用大量資源的便捷方法。

常見的RESOURCES更新考量

以下所有端點都是關於更新的JICSRESOURCES。使用該groupName字段，您可以更改任何GROUP的擁有 JICSRESOURCE，前提是字段值指向基礎JICS存儲GROUP中的現有（否則，您將從端點獲得BADREQUEST響應（HTTPSTATUS400））。

更新一個 TRANSACTION

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：/api/services/rest/jicsservice/updateTransaction
- 引數：有JSON效載荷，代表JICSTRANSACTION要更新。這是一個com.netfective.bluage.jac.entities.JACTransaction對象的JSON序列化。
- 返回一個布爾值。如果值為「true」，表示TRANSACTION已在基礎JICS儲存區中成功更新。

更新一個 PROGRAM

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：/api/services/rest/jicsservice/updateProgram
- 引數：有JSON效載荷，代表JICSPROGRAM要更新。這是一個com.netfective.bluage.jac.entities.JACProgram對象的JSON序列化。
- 返回一個布爾值。如果值為「true」，表示PROGRAM已在基礎JICS儲存區中成功更新。

更新一個 FILE

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：/api/services/rest/jicsservice/updateFile
- 引數：有JSON效載荷，代表JICSFILE要更新。這是一個com.netfective.bluage.jac.entities.JACFile對象的JSON序列化。
- 返回一個布爾值。如果值為「true」，表示FILE已在基礎JICS儲存區中成功更新。

更新一個 TDQUEUE

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：`/api/services/rest/jicsservice/updateTDQueue`
- 引數：有JSON效載荷，代表JICSTDQUEUE要更新。這是一個`com.netfective.bluage.jac.entities.JACTDQueue`對象的JSON序列化。
- 返回一個布爾值。如果值為「true」，表示TDQueue已在基礎JICS儲存區中成功更新。

更新一個 TSMODEL

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：`/api/services/rest/jicsservice/updateTSModel`
- 引數：有JSON效載荷，代表JICSTSMODEL要更新。這是一個`com.netfective.bluage.jac.entities.JACTSModel`對象的JSON序列化。
- 返回一個布爾值。如果值為「true」，表示TSMODEL已在基礎JICS儲存區中成功更新。

更新圖元

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：`/api/services/rest/jicsservice/updateElements`
- 引數：代表要更新之元素的JSON裝載。
- 返回一個布爾值，其中`true`表示元素更新已在基礎JICS存儲中成功操作。

Upsert 元素

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：`/api/services/rest/jicsservice/upsertElements`
- 引數：代表要提升元素的JSON有效負載。
- 返回一個布爾值，其中`true`表示元素 upsert 在基礎JICS存儲器中成功操作。

擷取元素

- 支持的方法：GET
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：`/api/services/rest/jicsservice/retrieveElements`
- 引數：無
- 傳回所有序列化JICS資源的清單。

JICSCRUD操作

- 支持的方法：POST
- 需要驗證和下列其中一個角色：ROLEROLE_ ADMIN、SUPER _ ADMIN、ROLE _ USER
- 路徑：`/api/services/rest/jicsservice/jicsCrudOperation`
- 參數：代表您正在尋找的JICS資源的JSON有效載荷。這是一個`com.netfective.bluage.jac.entities.request.JicsCrudOperationRequest`對象的JSON序列化。
- 返回代表響應的JSON有效載荷。這是一個`com.netfective.bluage.jac.entities.request.JicsCrudOperationResponse`對象的JSON序列化。

其他

主題

- JICS伺服器健康狀態

JICS伺服器健康狀態

- 支持的方法：GET
- 路徑：`/api/services/rest/jicsserver/serverIsUp`
- 引數：無
- 返回：無。HTTPSTATUS200 回應表示伺服器已啟動並執行中。

JAC使用者管理端點

使用下列端點來管理使用者互動。

主題

- 記錄使用者
- 測試系統中是否至少存在用戶
- 錄製新使用者
- 使用者資訊
- 列出使用者
- 刪除使用者
- 登出目前的使用者

記錄使用者

- 支持的方法：POST
- 路徑：`/api/services/security/servicelogin/login`
- 引數：無
- 返回對`com.netfective.bluage.jac.entities.SignOn`象的JSON序列化，代表其憑據在當前請求中提供的用戶。密碼會從傳回物件的檢視中隱藏起來。指定給使用的角色正在列出。

回應範例：

```
{
    "login": "some-admin",
    "password": null,
    "roles": [
      {
        "id": 0,
        "roleName": "ROLE_ADMIN"
      }
    ]
  }
```

測試系統中是否至少存在用戶

- 支持的方法：GET

- 路徑：/api/services/security/servicelogin/hasAccount
- 引數：無
- true如果至少已建立預設超級管理員使用者以外的其他使用者，則傳回布林值。false否則返回。

錄製新使用者

- 支持的方法：POST
- 需要驗證和 ROLE _ ADMIN 角色。
- 路徑：/api/services/security/servicelogin/recorduser
- 參數：對com.netfective.bluage.jac.entities.SignOn象的JSON序列化，代表要添加到存儲的用戶。應定義使用者的角色，否則使用者可能無法使用該JAC設施和端點。
- true如果成功創建用戶，則返回布爾值。false否則返回。

樣品請求：

```
{
    "login": "simpleuser",
    "password": "simplepassword",
    "roles": [
      {
        "id": 2,
        "roleName": "ROLE_USER"
      }
    ]
  }
```

錄製新使用者時，只能使用下列角色：

- ROLE_ADMIN：可以管理JICS資源和用戶。
- ROLE_USER：可以管理JICS資源，但不能管理使用者。

使用者資訊

- 支持的方法：GET
- 路徑：/api/services/security/servicelogin/userInfo
- 引數：無

- 返回當前連接的用戶的用戶名和角色。

列出使用者

- 支持的方法：GET
- 需要驗證和 ROLE _ ADMIN 角色。
- 路徑：`/api/services/security/servicelogin/listusers`
- 引數：無
- 返回一個列表`com.netfective.bluage.jac.entities.SignOn`，序列化為JSON。

刪除使用者

- 支持的方法：POST
- 需要驗證和 ROLE _ ADMIN 角色。
- 路徑：`/api/services/security/servicelogin/deleteuser`
- 參數：代表要從存儲中刪除的用戶的`com.netfective.bluage.jac.entities.SignOn`對象的JSON序列化。
- 返回布爾值，`true`如果用戶被成功刪除。

> **Important**
>
> 這個操作無法復原。刪除的使用者將無法再次連線至JAC應用程式。

登出目前的使用者

- 支持的方法：GET
- 路徑：`/api/services/security/servicelogout/logout`
- 引數：無
- `{"success":true}`如果目前使用者已成功登出，則傳回JSON訊息。相關HTTP工作階段將失效。

AWS 藍光時代用戶的數據結構

您可以在以下部分中了解 AWS Blu Age 引擎的各種數據結構。

主題

- Job 執行詳細訊息結構
- 異動啟動結果結構
- 異動啟動記錄結果結構
- 佇列中工作的可能狀態
- 提交工作和排程工作輸入
- 已排程的工作回應清單
- 重複工作回應清單

Job 執行詳細訊息結構

每個作業執行詳細資訊都會包含下列欄位：

scriptId

被調用腳本的標識符。

來電者

來電者的 IP 地址。

identifier

唯一的作業執行識別碼。

startTime

工作執行開始的日期和時間。

endTime

工作執行結束的日期和時間。

status

工作執行的狀態。其中一個可能的值：

- DONE: 工作執行正常結束。
- TRIGGERED：工作執行已觸發但尚未啟動。
- RUNNING：工作執行正在執行。
- KILLED：工作執行已終止。

- FAILED: 工作執行失敗。

executionResult

總結作業執行結果的消息。如果工作執行尚未完成，則此訊息可以是簡單訊息，也可以是具有下列欄位的JSON結構：

- exitCode: 數字結束代碼；負值表示失敗情況。
- 程序：由工作推出的最新程序。
- status：其中一個可能的值：
  - Error: 當 exitCode = -1；這對應於工作執行期間發生的 (技術) 錯誤。
  - Failed: 當 exitcode = -2; 這對應於服務程序執行期間發生的故障（如ABEND情況）。
  - Succeeded：當 exitCode > = 0 時；
- stepName：工作中執行的最新步驟名稱。

executionMode

SYNCHRONOUS或者ASYNCHRONOUS，取決於工作啟動的方式。

輸出範例：

```
{
    "scriptId": "INTCALC",
    "caller": "127.0.0.1",
    "identifier": "97d410be-efa7-4bd3-b7b9-d080e5769771",
    "startTime": "06-09-2023 11:42:41",
    "endTime": "06-09-2023 11:42:42",
    "status": "DONE",
    "executionResult": "{ \"exitCode\": -1, \"stepName\": \"STEP15\", \"program\":
 \"CBACT04C\", \"status\": \"Error\" }",
    "executionMode": "ASYNCHRONOUS"
  }
```

異動啟動結果結構

結構可能包含下列欄位：

outCome

表示事務執行結果的字符串。可能值為：

- Success：事務執行正確到底。

- `Failure`: 事務執行無法正常結束，遇到一些問題。

同志群

代表COMMAREA最終值的字串，做為 byte64 編碼的位元組陣列。可能是一個空字符串。

containerRecord

(選擇性) 字串，將CONTAINER記錄內容表示為 byte64 編碼位元組陣列。

serverDescription

可能包含有關服務請求的服務器的信息（用於調試目的）。可能是一個空字符串。

abendCode

(選擇性) 如果已啟動交易所參照的程式異常結束，異常結束代碼值會在此欄位中以字串形式傳回。

範例回應：

Success (成功)

```
{
    "outCome": "Success",
    "commarea": "",
    "serverDescription": ""
  }
```

失敗

```
{
    "outCome": "Failure",
    "commarea": "",
    "serverDescription": "",
    "abendCode": "AEIA"
  }
```

異動啟動記錄結果結構

結構可能包含下列欄位：

recordContent

將記錄內容表示為 byte64 編碼位元組陣列COMMAREA的字串。

containerRecord

將記錄內容表示為 byte64 編碼位元組陣列CONTAINER的字串。

serverDescription

可能包含有關服務請求的服務器的信息（用於調試目的）。可能是一個空字符串。

範例回應：

Success (成功)

```
{
    "recordContent": "",
    "serverDescription": ""
}
```

佇列中工作的可能狀態

在佇列上，工作可以具有下列狀態：

ACTIVE

工作目前正在佇列上執行。

EXECUTION_WAIT

工作正在等待執行緒可用。

SCHEDULED

工作排定在特定日期和時間執行。

HOLD

在執行之前，Job 正在等待釋放。

COMPLETED

Job 已成功執行。

FAILED

Job 執行失敗。

UNKNOWN

狀態是未知的。

提交工作和排程工作輸入

提交工作和排程工作輸入
是com.netfective.bluage.gapwalk.rt.jobqueue.SubmitJobMessage物件的JSON序列化。下面的示例輸入顯示了這樣一個 bean 的所有字段。

提交工作的樣本輸入：

```
{
    "messageQueueName":null,
    "scheduleDate":null,
    "scheduleTime":null,
    "programName":"PTA0044",
    "programParams":
     {"wmind":"B"},
    "localDataAreaValue":"",
    "userName":"USER1",
    "jobName":"PTA0044",
    "jobNumber":9,
    "jobPriority":5,
    "executionDate":"20181231",
    "jobQueue":"queue1",
    "jobOnHold":false
}
```

排程工作的輸入範例：

```
{
     "scheduleCron": "*/2 * * * * ?",
     "programName":"LOGPGM",
     "programParams": {
         "cl_sbmjob_param_json": "[\"./output/schedule-job-log.txt\", \"Every 2
 seconds!\"]"
     },
     "localDataAreaValue":"",
     "userName":"PVO",
     "jobName":"LOGGERJOB",
     "jobPriority":5,
     "jobQueue":"queue1",
     "scheduleMisfirePolicy": 4,
     "startTime": "2003/05/04 07:00:00.000 GMT-06:00",
     "endTime": "2003/05/04 07:00:07.000 GMT-06:00"
```

```
}
```

jobNumber

如果作業編號為 0，將使用作業編號序列中的下一個號碼自動生成作業編號。該值應設置為 0（除了測試目的）。

jobPriority

AS400 中的預設工作優先順序為 5。有效範圍為 0-9，0 表示最高優先順序。

jobOnHold

如果作業被保留提交，它不會立即執行，但只有當有人「釋放」它時才會執行。工作可以使用 REST API (/釋出或 /發行-全部) 來釋放。

scheduleDate 和 scheduleTime

如果這些值不為 null，工作將在指定的日期和時間執行。

日期

可以提供格式MMddyy或 ddMMyyyy （輸入的大小將決定使用什麼格式）

時間

可以提供格式HHmm或HHmmss（輸入的大小將決定使用什麼格式）

programParams

將作為地圖傳遞給程序。

scheduleMisfirePolicy

定義觸發錯誤時使用的策略。以下為可能值：

1. 釋放第一次失火並丟棄其他失火。
2. 為第一次失火提交暫停工作，並丟棄其他失火。
3. 丟棄失火。
4. 釋放所有的失火。工作佇列會執行所有工作。

已排程的工作回應清單

這是清單工作工作佇列端點的結構。用來送出該工作的送出工作訊息是回應的一部分。這可用於跟踪或測試/重新提交的目的。工作完成後，也會填入開始日期和結束日期。

```
[
  {
    "jobName": "PTA0044",
    "userName": "USER1",
    "jobNumber": 9,
    "jobPriority": 5,
    "status": "HOLD",
    "jobDelay": 0,
    "startDate": null,
    "endDate": null,
    "jobQueue": "queue1",
    "message": {
      "messageQueueName": null,
      "scheduleDate": null,
      "scheduleTime": null,
      "programName": "PTA0044",
      "programParams": {"wmind": "B"},
      "localDataAreaValue": "",
      "userName": "USER1",
      "jobName": "PTA0044",
      "jobNumber": 9,
      "jobPriority": 5,
      "executionDate": "20181231",
      "jobQueue": "queue1",
      "jobOnHold": true,
      "scheduleCron": null,
      "save": false,
      "scheduleMisfirePolicy": 4,
      "omitdates": null
    },
    "executionId": 1,
    "jobScheduledId": 0,
    "jobScheduledAt": null
  },
  {
    "jobName": "PTA0044",
    "userName": "USER1",
    "jobNumber": 9,
    "jobPriority": 5,
    "status": "COMPLETED",
    "jobDelay": 0,
    "startDate": "2022-10-13T22:48:34.025+00:00",
    "endDate": "2022-10-13T22:52:54.475+00:00",
```

```
    "jobQueue": "queue1",
    "message": {
      "messageQueueName": null,
      "scheduleDate": null,
      "scheduleTime": null,
      "programName": "PTA0044",
      "programParams": {"wmind": "B"},
      "localDataAreaValue": "",
      "userName": "USER1",
      "jobName": "PTA0044",
      "jobNumber": 9,
      "jobPriority": 5,
      "executionDate": "20181231",
      "jobQueue": "queue1",
      "jobOnHold": true,
      "scheduleCron": "*/20 * * * * ?",
      "save": false,
      "scheduleMisfirePolicy": 4,
      "omitdates": null
    },
    "executionId": 2,
    "jobScheduledId": 0,
    "jobScheduledAt": null
  }
]
```

重複工作回應清單

這是/排程/清單工作佇列端點的結構。

```
[
  {
    "id": 1,
    "status": "ACTIVE",
    "jobNumber": 1,
    "userName": "PVO",
    "msg": {
      "messageQueueName": null,
      "scheduleDate": null,
      "scheduleTime": null,
      "startTime": "2024/03/07 21:12:00.000 UTC",
      "endTime": "2024/03/07 21:13:59.000 UTC",
      "programName": "LOGPGM",
```

```
      "programParams": {"cl_sbmjob_param_json": "[\"./output/schedule-job-log.txt\",
\"Every 20 seconds!\"]"},
      "localDataAreaValue": "",
      "userName": "PVO",
      "jobName": "LOGGERJOB",
      "jobNumber": 1,
      "jobScheduleId": 1,
      "jobPriority": 5,
      "executionDate": null,
      "jobQueue": "queue1",
      "jobOnHold": false,
      "scheduleCron": "*/20 * * * * ?",
      "save": false,
      "scheduleMisfirePolicy": 4,
      "omitdates": null
    },
    "lastUpdatedAt": "2024-03-07T21:11:13.282+00:00",
    "lastUpdatedBy": ""
  }
]
```

設置 AWS 藍光時代運行時（非託管）

本節介紹了在 AWS 基礎架構上設置 AWS Blu Age 運行時（非託管）的步驟。在為應用程式設定 AWS Blu Age Runtime (非託管) 之前，請先瞭解必要條件、區域和值區，以及用來設定和管理執行階段環境的 CloudWatch警示設定。

主題

- AWS 藍色年齡運行時先決
- 入職 AWS 藍光時代運行時
- AWS Blu Age 運行時的基礎架構設置要求（非託管）
- 在 Amazon 上部署 AWS 藍色時代運行時ECS由 AWS Fargate
- 在 Amazon 上部署 AWS 藍光時代運行 EC2
- 測試 PlanetsDemo 應用程式

AWS 藍色年齡運行時先決

AWS Blu 年齡運行時（非託管）在多個the section called “AWS 藍光時代版本資訊”發行版本中提供。如果您有正在進行的現代化專案，則可能需要執行階段的增量版本以進行實作和測試。要定義您的需求，請聯繫您的 AWS Blu Oge 送貨經理。

在開始 AWS Blu Oge 執行階段 (非託管) 入職程序之前，請執行下列動作：

- 確保您有一個 AWS 帳戶。
- 確保您使用 Blu 時代重構了現代化的應用程序。 AWS
- 選擇一個 AWS 區域和 AWS Blu 年齡運行時（非託管）支持的計算選項之一。
- 選擇您要使用的 AWS 藍色時代運行時版本。
- 審查the section called “基礎架構設定需”並驗證運行 AWS Blu 時代運行時（非託管）所需的其他組件。

> **Note**
>
> 如果要測試 AWS 藍光時代運行時（非託管）的功能，則可以使用演示應用程序`Planets Demo`，您可以從 PlanetsDemo-v1.zip 下載。

入職 AWS 藍光時代運行時

要開始使用，請創建一個 AWS Support 案例以請求入職以訪問 AWS Blu Age 運行時。在您的請求中包含您的 AWS 帳戶 ID、您要使用的 AWS 區域，以及計算選項和執行階段版本。如果您不確定需要哪個版本，請聯繫您的 AWS Blu Age 送貨經理。

> **Note**
>
> AWS 藍光時代運行時有兩種主要品種：alpha 預發布和官方發布。要確定要使用哪個版本，請參閱 Blu Insights 網站上的入門，或聯繫您的 AWS Blu Age 交付經理。

Amazon 上的區域和存儲桶 AWS 藍光時代運行時（非託管）EC2

我們會依區域和運算選項，將 AWS Blu Age 執行階段 (非受管) 成品存放在不同的 Amazon S3 儲存貯體中。要在 Amazon 上訪問 AWS Blu Age 運行時（非託管）的存儲桶EC2，請使用下表中列出的名稱。 AWS 區域

> **Note**
>
> 此表適用於 Amazon EC2 以及 Amazon ECS 和 EC2 Amazon 中使用的 Amazon 實例EKS。

AWS 區域	釋放桶	預先釋放鏟斗
美國東部 (俄亥俄)	aws-bluage-runtime-artifact sus-east-2	aws-bluage-runtime-artifacts-us-east-2
美國東部 (維吉尼亞北部)	aws-bluage-runtime-artifact sus-east-1	aws-bluage-runtime-artifacts-開發-美國東部 1
美國西部 (加利佛尼亞北部)	aws-bluage-runtime-artifact s-788454540782-us-west-1	aws-bluage-runtime-artifacts-開發 -78845404878-us-west-1
美國西部 (奧勒岡)	aws-bluage-runtime-artifact s-836771190483-us-west-2	aws-bluage-runtime-artifact s-開發 -836771190483-us-west-2
歐洲 (愛爾蘭)	aws-bluage-runtime-artifact s-92527819477 eu-west-1	aws-bluage-runtime-artifacts-開發 -9252781947 eu-west-1
Europe (Paris)	aws-bluage-runtime-artifact s-673009995881 eu-west-3	aws-bluage-runtime-artifact s-開發 -673009995881 eu-west-3
歐洲 (法蘭克福)	aws-bluage-runtime-artifact s-485196800481 eu-central-1	aws-bluage-runtime-artifacts-開發-eu-central-1
南美洲 (聖保羅)	aws-bluage-runtime-artifact s-737536804457-sa-east-1	aws-bluage-runtime-artifacts-開發 -73753680457-sa-east-1

AWS 區域	釋放桶	預先釋放鏟斗
亞太區域 (東京)	aws-bluage-runtime-artifacts-445578176276-ap-northeast-1	aws-bluage-runtime-artifacts-445578176276-ap-northeast-1
亞太區域 (悉尼)	aws-bluage-runtime-artifacts-726160321909 ap-southeast-2	aws-bluage-runtime-artifacts-開發 -726160321909 ap-southeast-2

區域和存儲桶 AWS 藍光時代運行時（非託管）在 Amazon 上由 Fargate ECS 管理

我們會依區域和運算選項，將 AWS Blu Age 執行階段 (非受管) 成品存放在不同的 Amazon S3 儲存貯體中。要在 Fargate 管理的 Amazon 上訪問 AWS Blu 年齡運行時（非ECS託管）的存儲桶，請使用下表中列出的名稱。 AWS 區域

AWS 區域	釋放桶	預先釋放鏟斗
美國東部 (俄亥俄)	aws-bluage-runtime-fargate-美國東方 -2	aws-bluage-runtime-fargate-開發-us-east-2
美國東部 (維吉尼亞北部)	aws-bluage-runtime-fargate-美國東方一號	aws-bluage-runtime-fargate-美國東部第一
美國西部 (加利佛尼亞北部)	aws-bluage-runtime-fargate-美國西部 -1	aws-bluage-runtime-fargate-美國西部 1
美國西部 (奧勒岡)	aws-bluage-runtime-fargate-us-west-2	aws-bluage-runtime-fargate-開發 -688933007849-us-west-2
歐洲 (愛爾蘭)	aws-bluage-runtime-fargate-歐洲西部 -1	aws-bluage-runtime-fargate-開發 -140138033705 eu-west-1
Europe (Paris)	aws-bluage-runtime-fargate-eu-west-3	aws-bluage-runtime-fargate-33971294811 eu-west-3
歐洲 (法蘭克福)	aws-bluage-runtime-fargate-eu-central-1	aws-bluage-runtime-fargate-歐盟中央區 -1

AWS 區域	釋放桶	預先釋放鏟斗
南美洲 (聖保羅)	aws-bluage-runtime-fargate-相關 -767397998881-sa-east-1	aws-bluage-runtime-fargate-開發 -767397998881 sa-east-1
亞太區域 (東京)	aws-bluage-runtime-fargate-RL-ap-northeast-1	aws-bluage-runtime-fargate-開發 -891377400849-ap-northeast-1
亞太區域 (悉尼)	aws-bluage-runtime-fargate-阿 ap-southeast-2	aws-bluage-runtime-fargate-開發 -533267435478-ap-southeast-2

使 AWS CLI 用列出值區的內容

登入後，您可以在終端機中執行下列 AWS CLI 命令來列出值區的內容。

```
aws s3 ls bucket-name
```

bucket-name替換為上一個表格 AWS 區域 中的值區的名稱。

此命令會傳回與 AWS Blu Age 執行階段 (非受管理) 執行階段不同版本相對應的資料夾清單，例如下列發行值區：

```
PRE 3.10.0/
PRE 4.0.0/
```

或者構建存儲桶的以下內容：

```
PRE 4.1.0-alpha.8/
PRE 4.1.0-alpha.9/
```

我們建議您使用可用的最新版本。如果這是不可能的，那麼使用在應用程序重構階段驗證的運行時版本。若要列出特定版本的可用架構，請執行下列命令：

```
aws s3 ls s3://bucket-name/version/Framework/
```

將值區bucket-name的名稱取代為您version的 AWS 區域 和您想要的版本。以下是兩個範例。

若為核發時段：

```
aws s3 ls s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/4.0.0/
Framework/
```

該命令返回框架列表，例如：

```
2024-04-08 16:11:19  152040176 aws-bluage-runtime-4.0.0.tar.gz
2024-04-08 16:11:50         45 aws-bluage-runtime-4.0.0.tar.gz.checksumSHA256
2024-04-08 16:11:52  176518889 aws-bluage-webapps-4.0.0.tar.gz
2024-04-08 16:12:28         45 aws-bluage-webapps-4.0.0.tar.gz.checksumSHA256
```

對於構建桶：

```
aws s3 ls s3://aws-bluage-runtime-artifacts-dev-139023371234-us-
east-1/4.1.0-alpha.9/Framework/
```

該命令返回框架列表，例如：

```
2024-04-09 20:23:34  152304534 aws-bluage-runtime-4.1.0-alpha.9.tar.gz
2024-04-09 20:24:05         45 aws-bluage-runtime-4.1.0-alpha.9.tar.gz.checksumSHA256
2024-04-09 20:24:07  176262381 aws-bluage-webapps-4.1.0-alpha.9.tar.gz
2024-04-09 20:24:42         45 aws-bluage-webapps-4.1.0-alpha.9.tar.gz.checksumSHA256
```

下載框架

您可以下載框架，例如在現有的 Amazon EC2 實例上升級 AWS Blu Age 運行時版本。

```
aws s3 cp s3://bucket-name/version/Framework/ folder-of-your-choice --
recursive
```

其中：

folder-of-your-choice

您要下載框架的文件夾路徑。

例如：`aws s3 cp s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/4.0.0/Framework/ . --recursive`

此命令會產生以下的輸出：

```
download: s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/4.0.0/
Framework/aws-bluage-runtime-4.0.0.tar.gz.checksumSHA256 to ./aws-bluage-
runtime-4.0.0.tar.gz.checksumSHA256
download: s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/4.0.0/
Framework/aws-bluage-webapps-4.0.0.tar.gz.checksumSHA256 to ./aws-bluage-
webapps-4.0.0.tar.gz.checksumSHA256
download: s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/4.0.0/Framework/aws-
bluage-webapps-4.0.0.tar.gz to ./aws-bluage-webapps-4.0.0.tar.gz
download: s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/4.0.0/Framework/aws-
bluage-runtime-4.0.0.tar.gz to ./aws-bluage-runtime-4.0.0.tar.gz
```

您可以列出框架文件，如下所示：

```
ls -l
```

此命令會產生以下的輸出：

```
total 230928
-rw-rw-r-- 1 cloudshell-user cloudshell-user 152040176 Apr  8 16:11 aws-bluage-
runtime-4.0.0.tar.gz
-rw-rw-r-- 1 cloudshell-user cloudshell-user        45 Apr  8 16:11 aws-bluage-
runtime-4.0.0.tar.gz.checksumSHA256
-rw-rw-r-- 1 cloudshell-user cloudshell-user 176518889 Apr  8 16:11 aws-bluage-
webapps-4.0.0.tar.gz
-rw-rw-r-- 1 cloudshell-user cloudshell-user        45 Apr  8 16:12 aws-bluage-
webapps-4.0.0.tar.gz.checksumSHA256
```

AWS Blu Age 運行時的基礎架構設置要求（非託管）

本主題說明執行 AWS Blu Age 執行階段 (非受管理) 所需的最低基礎結構組態。以下程序描述了如何在您選擇的計算上設置 AWS Blu Age 運行時（非託管），以在 AWS Blu Age 運行時部署現代化的應用程序。您建立的資源必須位於具有專VPC用於您應用程式網域的子網路的 Amazon 中。

主題

- 基礎設施需求
- AWS 藍光時代運行時的 Amazon EC2 實例類型（在 Amazon 上EC2）
- 在 Amazon 上運行 AWS 藍光時代運行 EC2
- 在 Amazon 上運行 AWS 藍光時代運ECS行時 EC2

- 在 Amazon 上運行 AWS 藍光時代運EKS行時 EC2
- 在 Amazon 上運行 AWS 藍光時代運行時ECS由 AWS Fargate

基礎設施需求

建立安全群組

如果您打算在 Amazon 上使用 Amazon EC2 執行個體EKS，請略過此程序，因為 Amazon EKS 叢集建立程序會代表您建立安全群組。請在下列程序中使用該安全性群組，而非建立新群組。

1. 在打開 Amazon VPC 控制台https://console.aws.amazon.com/vpc/。
2. 在左側導覽窗格的 [安全性] 下，選擇 [安全性群組]。
3. 在中央窗格中，選擇 [建立安全性群組]。
4. 在 [安全性群組名稱] 欄位中，輸入**M2BluagePrivateLink-SG**。
5. 在 Inbound rules (傳入規則) 區段中，選擇 Add rule (新增規則)。
6. 對於「類型」，選擇HTTPS。
7. 針對來源，輸入您的 VPCCIDR.
8. 在 [輸出規則] 區段中，選擇 [新增規則]。
9. 對於「類型」，選擇HTTPS。
10. 針對 Destination (目標)，輸入 **0.0.0.0/0**。
11. 選擇建立安全群組。

創建一個 Amazon VPC 端點

1. 在打開 Amazon VPC 控制台https://console.aws.amazon.com/vpc/。
2. 在左側導覽窗格的「虛擬私人雲端」下，選擇「端點」。
3. 在中央窗格中，選擇「建立端點」。
4. 在「服務」區段中，**SQS**在搜尋欄位中輸入，然後選取與您的區域對應的 Amazon SQS 服務。
5. 在VPC此VPC區段中，選取您在上一個步驟中建立的 Amazon。
6. 在「子網路」區段中，選取您為應用程式網域建立的子網路。
7. 在 [安全性群組] 區段中，從上一個程序中選取安全性群組。
8. 選擇建立端點。

建立IAM策略

1. 在開啟IAM主控台https://console.aws.amazon.com/iam/。
2. 在左側導覽窗格的 [存取管理] 下，選擇 [原則]。
3. 在中央窗格中，選擇 [建立原則]。
4. 在 [原則編輯器] 區段中，選擇JSON。
5. 將您在編輯器中看JSON到的所有內容替換為以下內容JSON。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "sqs:GetQueueUrl",
                "sqs:ReceiveMessage",
                "sqs:SendMessage"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 如果您需要更多詳細資訊來自訂您的保單，請聯絡您的 AWS Blu Age 送貨經理或客戶經理。

6. 選擇 Next (下一步)。
7. 輸入策略的名稱，然後選擇「建立策略」。

建立 IAM 角色

1. 在開啟IAM主控台https://console.aws.amazon.com/iam/。
2. 在左側導覽窗格的 [存取管理] 下，選擇 [角色]。
3. 在中央窗格中，選擇 [建立角色]。
4. 在 [使用案例] 區段中，根據您的運算選擇，選擇下列其中一項：

- EC2（適用於 Amazon EKS 上的 Amazon EC2 和 AmazonEC2）
- 彈性容器服務，然後是彈性容器服務的EC2角色（適用於 Amazon ECS 上的 AmazonEC2）
- 彈性容器服務，然後是彈性容器服務任務（適用於 Fargate ECS 管理的 Amazon）

5. 選擇 Next (下一步)。
6. 在搜尋方塊中，輸入您先前建立的原則名稱。
7. 選取政策左側的核取方塊。

> **Note**
>
> 如果您無法新增原則，請完成建立角色，然後更新角色以新增原則。

8. 選擇 Next (下一步)。
9. 輸入角色名稱，然後選擇建立角色。

AWS 藍光時代運行時的 Amazon EC2 實例類型（在 Amazon 上EC2）

以下是創建 Amazon EC2 實例或定義 Amazon EKS 工作者節點時可用於 AWS Blu Age 運行時（在 Amazon EC2 上EC2）的 Amazon 實例類型的列表。

```
t3.xlarge
t3.small
t3.large
t2.small
t2.large
r7a.medium
r7a.large
r7a.xlarge
r7a.2xlarge
r7a.4xlarge
r7a.8xlarge
r7a.12xlarge
r7a.16xlarge
r7a.24xlarge
r7a.32xlarge
r7a.48xlarge
r7a.metal-48xl
r7i.large
r7i.xlarge
```

```
r7i.2xlarge
r7i.4xlarge
r7i.8xlarge
r7i.12xlarge
r7i.16xlarge
r7i.24xlarge
r7i.48xlarge
r7i.metal-24xl
r7i.metal-48xl
r6i.xlarge
r6i.large
r6i.4xlarge
r6i.2xlarge
r5b.xlarge
r5b.large
r5b.2xlarge
r3.xlarge
m6i.xlarge
m6i.large
m6i.8xlarge
m6i.4xlarge
m6i.2xlarge
m6i.16xlarge
m5zn.xlarge
m5zn.large
m5zn.3xlarge
m5zn.2xlarge
m5.xlarge
m5.large
m5.8xlarge
m5.4xlarge
m5.2xlarge
m5.16xlarge
m5.12xlarge
c6i.xlarge
c6i.large
c6i.8xlarge
c6i.4xlarge
c6i.2xlarge
c6i.16xlarge
c5.xlarge
c5.large
c5.9xlarge
c5.4xlarge
```

```
c5.2xlarge
c5.18xlarge
c5.12xlarge
```

在 Amazon 上運行 AWS 藍光時代運行 EC2

若要建立 Amazon EC2 執行個體，請使用下列步驟。

創建一個 Amazon EC2 實例

1. 在打開 Amazon EC2 控制台https://console.aws.amazon.com/ec2/。
2. 選擇啟動執行個體。
3. 對於執行個體類型，請選擇中列出的其中一種類型the section called “AWS 藍光時代運行時的 Amazon EC2 實例類型（在 Amazon 上EC2）”。
4. 在 [key pair] 區段中，選擇現有金鑰配對或建立新金鑰配對。
5. 在 [網路設定] 區段中，選擇 [選取現有安全性群組]。
6. 對於「一般」安全性群組，請選擇 M2 BluagePrivateLink-SG。
7. 展開 [進階詳細資料] 區段。
8. 對於IAM執行個體設定檔，請選擇您先前建立的IAM角色。
9. 選擇啟動執行個體。

在 Amazon EC2 執行個體上安裝應用程式

1. Amazon EC2 執行個體的狀態變更為執行中時，請連線至執行個體。
2. 在執行個體上安裝下列軟體元件：

   - Java 運行時環境（JRE）17.
   - 阿帕奇雄貓 10。
   - AWS 藍光時代運行時（在 Amazon 上EC2）。在 Apache Tomcat 安裝文件夾的根目錄下安裝 AWS 藍光時代運行時（某些文件將被添加，而其他文件將被覆蓋）。

   要安裝與 AWS Blu Age 運行時歸檔一起提供的其他 Web 應用程序，請設置 Apache Tomcat 服務器的輔助實例，並在該位置解壓縮 web 應用程序存檔。

在 Amazon 上運行 AWS 藍光時代運ECS行時 EC2

1. 建立 Amazon ECS 叢集，並將 Amazon EC2 執行個體做為基礎基礎設施。請參閱 Amazon 彈性容器服務開發人員指南EC2中的 Amazon 上的 Windows 入門。
2. 指定您IAM在先前步驟中建立的角色。
3. 選擇中列出的其中一個執行個體類型the section called “AWS 藍光時代運行時的 Amazon EC2 實例類型（在 Amazon 上EC2）”。
4. 在 Amazon EC2 執行個體的網路設定中，選擇您在先前步驟中建立的安全群組。

在 Amazon 上運行 AWS 藍光時代運EKS行時 EC2

1. 創建一個 Amazon EKS 群集。請參閱 Amazon 使EKS用者指南中的建立 Amazon EKS 叢集。
2. 如前所述，會代表您建立安全性群組。您可以在建立 Amazon VPC 端點時使用該安全群組。
3. 建立節點群組。指定您IAM在先前步驟中建立的角色。
4. 選擇中列出的其中一個執行個體類型the section called “AWS 藍光時代運行時的 Amazon EC2 實例類型（在 Amazon 上EC2）”。
5. Amazon EKS 將自動將安全組分配給產生的 Amazon EC2 實例。

在 Amazon 上運行 AWS 藍光時代運行時ECS由 AWS Fargate

使用 AWSFargate (無伺服器) 做為基礎基礎設施來建立 Amazon ECS 叢集。請參閱 Amazon 彈性容器服務開發人員指南中的《開始使用 Fargate》。

在 Amazon 上部署 AWS 藍色時代運行時ECS由 AWS Fargate

您可以使用本節中的主題來了解如何在 Amazon 上設定 AWS Blu Age Runtime ECS 管理 AWS Fargate、如何更新執行階段版本、如何使用 Amazon CloudWatch 警示來監控部署，以及如何新增授權相依性。可以通過重建和重新部署 Docker 映像來實現升級到新的 AWS Blu Age 運行時版本。此外，您可以設定 Amazon CloudWatch 警示來監控應用程式日誌並接收錯誤通知。對於需要 Oracle 資料庫或 IBM MQ 等授權相依性的應用程式，您可以在 Docker 映像中包含必要的JAR檔案，並設定適當的設定。

主題

- 在 Amazon 上設置 AWS 藍光時代運行時由ECS管理 AWS Fargate
- 升級由 Amazon ECS 管理的 AWS 藍光時代運行時 AWS Fargate

- 在 Amazon 上為 AWS 藍光時代運行時設置 ECS Amazon CloudWatch 警報 AWS Fargate
- 在 Amazon ECS 上的 AWS Blu Age 運行時中設置許可依賴關係 AWS Fargate

在 Amazon 上設置 AWS 藍光時代運行時由ECS管理 AWS Fargate

本主題說明如何使用ECS管理的 Amazon 上的 AWS Blu Age 執行階段來設定和部署 PlanetsDemo 範例應用程式 AWS Fargate。

AWS 由 Amazon ECS 管理的藍色時代運行時可用 AWS Fargate 於 Linux/x86。

主題

- 必要條件
- 設定
- 測試部署的應用程式

必要條件

開始之前，請確定您已完成下列先決條件。

- AWS CLI 按照配置中的步驟配置 AWS CLI.
- 完成the section called “AWS 藍色年齡運行時先決”和the section called “入職 AWS 藍光時代運行時 ”。
- 在 Amazon 上下載由 AWS Fargate 二進製文件ECS管理的 B AWS lu Age 運行時。如需說明，請參閱 the section called “入職 AWS 藍光時代運行時 ”。
- 下載阿帕奇湯姆貓 10 二進製文件。
- 下載PlanetsDemo應用程式封存檔。
- 創建一個 Amazon Aurora Postgre SQL 數據庫JICS，並在其上運行`PlanetsDemo-v1/jics/sql/initJics.sql`查詢。如需如何建立 Amazon Aurora Postgre 資SQL料庫的相關資訊，請參閱建立並連線至 Aurora Postgre 資料SQL庫叢集。

設定

若要設定 PlanetsDemo 範例應用程式，請完成以下步驟。

1. 下載 Apache 的 Tomcat 二進製文件後，提取內容，然後轉到該`conf`文件夾。開啟要編輯的`catalina.properties`檔案，並取代以下行開`common.loader`頭的行。

```
common.loader="${catalina.base}/lib","${catalina.base}/lib/
*.jar","${catalina.home}/lib","${catalina.home}/lib/*.jar","${catalina.home}/
shared","${catalina.home}/shared/*.jar","${catalina.home}/extra","${catalina.home}/
extra/*.jar"
```

2. 通過使用 tar 命令來構建一個 `tar.gz` 歸檔壓縮 Apache 的 Tomcat 文件夾。
3. 準備一個 Docker 文件，以根據提供的運行時二進製文件和 Apache Tomcat 服務器二進製文件構建自定義映像。請參閱下面的例子碼頭文件。我們的目標是安裝 Apache 的 Tomcat 10，其次是 AWS藍光時代運行時（由 Amazon ECS 管理 AWS Fargate）提取在 Apache Tomcat 10 安裝目錄的根目錄下，然後安裝命名的示例現代化應用程序。 PlanetsDemo

> **Note**
>
> install-gapwalk.sh 和 install-app.sh 腳本，這是在這個例子碼頭文件中使用的內容，碼頭文件後列出。

```
FROM --platform=linux/x86_64 amazonlinux:2

RUN mkdir -p /workdir/apps
WORKDIR /workdir
COPY install-gapwalk.sh .
COPY install-app.sh .
RUN chmod +x install-gapwalk.sh
RUN chmod +x install-app.sh

# Install Java and AWS CLI v2-y
RUN yum install sudo java-17-amazon-corretto unzip tar -y
RUN sudo yum remove awscli -y
RUN curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
 "awscliv2.zip"
RUN sudo unzip awscliv2.zip
RUN sudo ./aws/install

#·Installation dir
RUN mkdir -p /usr/local/velocity/installation/gapwalk
# Copy PlanetsDemo archive to a dedicated apps dir
COPY PlanetsDemo-v1.zip /workdir/apps/

# Copy resources (tomcat, blu age runtime) to installation dir
```

```
COPY tomcat.tar.gz /usr/local/velocity/installation/tomcat.tar.gz
COPY aws-bluage-on-fargate-runtime-4.x.x.tar.gz /usr/local/velocity/installation/
gapwalk/gapwalk-bluage-on-fargate.tar.gz

# run relevant installation scripts
RUN ./install-gapwalk.sh
RUN ./install-app.sh

EXPOSE 8080
EXPOSE 8081
# ...

# Run Command to start Tomcat server
CMD ["sh", "-c", "sudo /bluage-on-fargate/tomcat.gapwalk/velocity/startup.sh
 $ECS_CONTAINER_METADATA_URI_V4 $AWS_CONTAINER_CREDENTIALS_RELATIVE_URI"]
```

以下是 install-gapwalk.sh 的內容。

```
#!/bin/sh

# Vars
TEMP_DIR=/bluage-on-fargate/tomcat.gapwalk/temp

# Install
echo "Installing Gapwalk and Tomcat"
sudo rm -rf /bluage-on-fargate
mkdir -p ${TEMP_DIR}
# Copy Blu Age runtime and tomcat archives to temporary extraction dir
sudo cp /usr/local/velocity/installation/gapwalk/gapwalk-bluage-on-fargate.tar.gz
 ${TEMP_DIR}
sudo cp /usr/local/velocity/installation/tomcat.tar.gz ${TEMP_DIR}
#·Create velocity dir
mkdir -p /bluage-on-fargate/tomcat.gapwalk/velocity
#·Extract tomcat files
tar -xvf ${TEMP_DIR}/tomcat.tar.gz -C ${TEMP_DIR}
# Copy all tomcat files to velocity dir
cp -fr ${TEMP_DIR}/apache-tomcat-10.x.x/* /bluage-on-fargate/tomcat.gapwalk/
velocity
# Remove default webapps of Tomcat
rm -f /bluage-on-fargate/tomcat.gapwalk/velocity/webapps/*
# Extract Blu Age runtime at velocity dir
tar -xvf ${TEMP_DIR}/gapwalk-bluage-on-fargate.tar.gz -C /bluage-on-fargate/
tomcat.gapwalk
```

```
# Remove temporary extraction dir
sudo rm -rf ${TEMP_DIR}
```

以下是 install-app.sh 的內容。

```
#!/bin/sh

APP_DIR=/workdir/apps
TOMCAT_GAPWALK_DIR=/bluage-on-fargate/tomcat.gapwalk

unzip ${APP_DIR}/PlanetsDemo-v1.zip -d ${APP_DIR}
cp -r ${APP_DIR}/webapps/* ${TOMCAT_GAPWALK_DIR}/velocity/webapps/
cp -r ${APP_DIR}/config/* ${TOMCAT_GAPWALK_DIR}/velocity/config/
```

4. 提供您在application-main.yml檔案中下列程式碼片段 (位於資料{TOMCAT_GAPWALK_DIR}/config夾中) 中作為先決條件之一部分所建立之資料庫的連線資訊。如需詳細資訊，請參閱建立並連線至 Aurora Postgre 資SQL料庫叢集。

```
datasource:
   jicsDs:
     driver-class-name :
     url:
     username:
     password:
     type :
```

5. 建立映像檔並將其推送至您的 Amazon ECR 儲存庫。如需指示，請參閱 Amazon 彈性容器登錄使用者指南中的推送 Docker 映像。
6. 在 https://console.aws.amazon.com/ecs/v2 開啟主控台。
7. 在左側導覽窗格中，選擇 [工作定義]。
8. 針對 [啟動類型]，選擇 AWS Fargate。
9. 選取您建立為其一部分的任務角色the section called “基礎架構設定需”。
10. 將您的圖像附加到容器。
11. 完成填寫表格，然後選擇 [建立]。
12. 在左側導覽窗格中，選擇 [叢集]，然後從清單中選擇您的叢集。
13. 在叢集的詳細資訊頁面上，選擇 [服務] 索引標籤上的 [建立]。
14. 選取作業定義。

15. 展開 [網路] 區段，然後VPC設定您建立為其中一部分的the section called “基礎架構設定需”子網路和安全性群組。
16. 部署您的 Amazon ECS 服務。

如果部署失敗，請檢查記錄檔。要找到它們，請轉到 Amazon ECS 管理的任務頁面 AWS Fargate，然後選擇日誌選項卡。如果您發現以 C 開頭的錯誤碼，然後是數字，例如CXXXX，請注意錯誤訊息。例如，錯誤碼 C5102 是表示基礎結構組態不正確的常見錯誤。您還可以在正在運行的任務中導航並運行一些命令，類似於 AWS Blu Age 運行時（在 Amazon 上EC2）。如需詳細資訊，請參閱 Amazon 彈性容器服務開發人員指南中的使用 Amazon ECS Exec 進行偵錯。

若要開啟互動式 Shell，請從本機電腦執行下列命令。

```
aws ecs execute-command --cluster your_cluster_name --container your_container_name --
task task_id --interactive --command /bin/sh
```

測試部署的應用程式

如需如何測試 PlanetsDemo 應用程式的範例，請參閱the section called “測試 PlanetsDemo 應用程式”。

升級由 Amazon ECS 管理的 AWS 藍光時代運行時 AWS Fargate

本指南介紹瞭如何升級由 Amazon ECS 管理的 AWS Blu Age 運行時 AWS Fargate。

主題

- 必要條件
- 升級 AWS 藍光時代運行時

必要條件

開始之前，請確定您符合下列先決條件。

- 完成the section called “AWS 藍色年齡運行時先決”和the section called “入職 AWS 藍光時代運行時”。
- 下載您要升級到的 AWS 藍光時代運行時的版本。如需詳細資訊，請參閱the section called “入職 AWS 藍光時代運行時 ”。該框架由兩個二進製文件組成：`aws-bluage-runtime-x.x.x.x.tar.gz`和`aws-bluage-webapps-x.x.x.x.tar.gz`。

升級 AWS 藍光時代運行時

請完成以下步驟來升級 AWS 藍光時代執行階段。

1. 使用所需的 AWS 藍光時代運行時版本重建您的 Docker 映像。如需說明，請參閱 the section called “在 Amazon 上設置 AWS 藍光時代運行時 ECS”。
2. 將您的 Docker 映像推送到您的 Amazon ECR 存儲庫。
3. 停止並重新啟動您的 Amazon ECS 服務。
4. 驗證記錄檔。

AWS 藍光時代運行時已成功升級。

在 Amazon 上為 AWS 藍光時代運行時設置 ECS Amazon CloudWatch 警報 AWS Fargate

您可以設 CloudWatch 定在部署的應用程式遇到例外狀況時收到更明顯的通知。這也可以幫助您接收應用程序日誌，並添加警報以警告您可能發生的錯誤。

警報設定

使用 CloudWatch 日誌，您可以根據應用程序和需求配置任意數量的指標和警報。

具體而言，您可以在建立 Amazon ECS 叢集期間直接為使用情況提醒設定主動警示，以便在發生錯誤時收到通知。要突出顯示與 AWS Blu Age 控制系統連接中的錯誤，請在日誌中添加有關字符串「錯誤 C」的指標。然後，您可以定義對此量度做出反應的警示。

在 Amazon ECS 上的 AWS Blu Age 運行時中設置許可依賴關係 AWS Fargate

本主題介紹如何設置其他許可依賴項，您可以在ECS管理的 Amazon 上與 AWS Blu Age 運行時一起使用 AWS Fargate。

主題

- 必要條件
- 概觀

必要條件

開始之前，請確定您已完成下列先決條件。

- 完成the section called “AWS 藍色年齡運行時先決”和the section called “入職 AWS 藍光時代運行時”。
- 從其來源取得下列相依性。

Oracle 資料庫

提供 Oracle 資料庫驅動程式。例如，一個例子 11 -二月三十三日.

IBMMQ 連接

提供 IBMMQ 用戶端。例如，請參閱 .im.mq. 加卡爾塔. 客戶端 9.3.4.1.jar.

使用此依賴版本，還提供以下傳遞依賴關係：

- 瓶裝百分之十五至十七
- BCPKIX-日本植物蛋白
- 布庫蒂爾-日本料十五至十八

DDS印表機檔案

提供賈斯珀報告庫。例如，茉莉報告 6.16.0.jar，但更新的版本可能是兼容的。

使用此依賴版本，還提供以下傳遞依賴關係：

- 田園核心 -1.4.1.jar
- 罐子-XML-1.4.1.罐
- 普通消化器 -2.1.1 罐
- 心形紅酒罐
- 項目 -2.1.7.J8.jar
- 噴射器 1. 罐
- 日本料理 -1.0.23 罐
- 自由圖 -1.0.19 罐
- 共同的無簷小豆 -1.9.4.jar
- 公用集合 -3.2.2.jar

概觀

若要安裝相依性，請完成以下步驟。

1. 根據需要將上述任何依賴項複製到 Docker 映像構建文件夾中。
2. 如果您的JICS或 Blusam 資料庫是在 Oracle 上託管，請在中提供 Oracle 資料庫驅動程式。*your-tomcat-path*/extra
3. 在您的 Dockerfile 上，將這些依賴關係複製到. *your-tomcat-path*/extra
4. 構建您的 Docker 映像，然後將其推送到 Amazon ECR。
5. 停止並重新啟動您的 Amazon ECS 服務。
6. 檢查日誌。

在 Amazon 上部署 AWS 藍光時代運行 EC2

您可以了解如何在 Amazon 上設定 AWS Blu Age Runtime (非受管) EC2、如何更新執行階段版本、如何使用 Amazon CloudWatch 警示來監控部署，以及如何使用本節中的主題新增授權相依性。當您創建 Amazon EC2 實例以及在 Amazon 上使用 Amazon EC2 或 Amazon ECS 上的 Amazon 時，這些說明都適用EC2。EKS

主題

- 在 Amazon 上設置 AWS 藍光時代運行時（非託管）EC2
- 在 Amazon ECS 和 Amazon EC2 上使用容器 EKS
- 升級 Amazon 上的 AWS 藍光時代運行時 EC2
- 設置 AWS 藍光時代運行時（在 Amazon 上EC2）Amazon CloudWatch 警報
- 在 Amazon 上的 AWS 藍光時代運行時中設置許可依賴項 EC2

在 Amazon 上設置 AWS 藍光時代運行時（非託管）EC2

本主題說明如何使用 Amazon EC2 上的 AWS Blu Age 執行階段 (非受管) 來設定和部署 PlanetsDemo 範例應用程式。

主題

- 必要條件
- 設定
- 測試部署的應用程式

必要條件

開始之前，請確定您已完成下列先決條件。

- AWS CLI 按照配置中的步驟配置 AWS CLI.
- 完成the section called “AWS 藍色年齡運行時先決”和the section called “入職 AWS 藍光時代運行時 ”。
- 使用其中一種支援的EC2執行個體類型建立 Amazon 執行個體。如需詳細資訊，請參閱開始使用 Amazon EC2 Linux 執行個體。
- 確保您可以成功連接到 Amazon EC2 執行個體，例如使用SSM.
- 下載並提取 AWS 藍光時代運行時 (在 Amazon 上EC2) 在*your-tomcat-path*/*. 請務必將`bluage.bin`檔案完全放在 Apache Tomcat 文件中 CATALINA_ HOME 和 CATALINA _ BASE 下所述的`CATALINA_HOME`環境變數所指定的位置。如需如何擷取 AWS Blu Age 執行階段的指示，請參閱the section called “入職 AWS 藍光時代運行時 ”。
- 下載PlanetsDemo應用程式封存檔。
- 解壓縮存檔，然後將應用程式上傳到您選擇的 Amazon S3 儲存貯體。
- 創建一個 Amazon Aurora Postgre SQL 數據庫JICS，並在其上運行`PlanetsDemo-v1/jics/sql/initJics.sql`查詢。如需如何建立 Amazon Aurora Postgre 資SQL料庫的相關資訊，請參閱建立並連線至 Aurora Postgre 資料SQL庫叢集。

設定

若要設定 PlanetsDemo 範例應用程式，請完成以下步驟。

1. Connect 到您的 Amazon EC2 實例，然後轉到您的 Apache Tomcat 10 安裝文件`conf`夾下的文件夾。開啟要編輯的`catalina.properties`檔案，並取代以下行開`common.loader`頭的行。

   ```
   common.loader="${catalina.base}/lib","${catalina.base}/lib/
   *.jar","${catalina.home}/lib","${catalina.home}/lib/*.jar","${catalina.home}/
   shared","${catalina.home}/shared/*.jar","${catalina.home}/extra","${catalina.home}/
   extra/*.jar"
   ```

2. 導覽至 *<your-tomcat-path>*/webapps 資料夾。
3. 使用以下 PlanetsDemo 命令從 Amazon S3 儲存貯體複製 PlanetsDemo-v1/Web應用程式s/ 資料夾中可用的二進位檔案。

   ```
   aws s3 cp s3://path-to-demo-app-webapps/ . --recursive
   ```

> **Note**
>
> 以先前解壓縮存檔URI的儲 PlanetsDemo 存貯體正確的 Amazon S3 取`path-to-demo-app-webapps`代。

4. 將`PlanetsDemo-v1/config/`資料夾的內容複製到`<your-tomcat-path>/config/`。
5. 提供您作為`application-main.yml`檔案下列程式碼片段中先決條件之一部分所建立之資料庫的連線資訊。如需詳細資訊，請參閱建立並連線至 Aurora Postgre 資SQL料庫叢集。

```
datasource:
   jicsDs:
     driver-class-name :
     url:
     username:
     password:
     type :
```

6. 啟動你的阿帕奇 Tomcat 服務器，並驗證日誌。

```
your-tomcat-path/startup.sh

tail -f your-tomcat-path/logs/catalina.log
```

如果您發現以 C 開頭的錯誤碼，然後是數字，例如CXXXX，請注意錯誤訊息。例如，錯誤碼 C5102 是表示基礎結構組態不正確的常見錯誤。

測試部署的應用程式

如需如何測試 PlanetsDemo 應用程式的範例，請參閱the section called “測試 PlanetsDemo 應用程式”。

在 Amazon ECS 和 Amazon EC2 上使用容器 EKS

本主題說明如何使用 Amazon 上的 AWS Blu Age 執行階段 (非受管) EC2 做為容器來設定和部署 PlanetsDemo 範例應用程式。

主題

- 必要條件

- 設定
- 測試部署的應用程式

必要條件

開始之前，請確定您已完成下列先決條件。

- AWS CLI 按照配置中的步驟配置 AWS CLI.
- 完成the section called “AWS 藍色年齡運行時先決”和the section called “入職 AWS 藍光時代運行時 ”。
- 下載 AWS 藍光時代運行時（在 Amazon 上EC2）。如需如何擷取執行階段的指示，請參閱the section called “入職 AWS 藍光時代運行時 ”。
- 下載PlanetsDemo應用程式封存檔。
- 創建一個 Amazon Aurora Postgre SQL 數據庫JICS，並在其上運行`PlanetsDemo-v1/jics/sql/initJics.sql`查詢。如需如何建立 Amazon Aurora Postgre 資SQL料庫的相關資訊，請參閱建立並連線至 Aurora Postgre 資料SQL庫叢集。

設定

若要設定 PlanetsDemo 範例應用程式，請完成以下步驟。

1. 準備一個 Docker 文件，以根據提供的運行時二進製文件和 Apache Tomcat 服務器二進製文件構建自定義映像。請參閱下面的例子碼頭文件。我們的目標是安裝 Apache 的 Tomcat 10，其次是 AWS 藍光時代運行時（在 Amazon 上EC2）提取在 Apache Tomcat 10 安裝目錄的根目錄，然後安裝命名的示例現代化應用程序。 PlanetsDemo在這個例子中使用的碼頭文件`install-gapwalk.sh`和`install-app.sh`腳本列在碼頭文件之後。

```
FROM --platform=linux/x86_64 amazonlinux:2

RUN mkdir -p /workdir/apps
WORKDIR /workdir
COPY install-gapwalk.sh .
COPY install-app.sh .
RUN chmod +x install-gapwalk.sh
RUN chmod +x install-app.sh

# Install Java and AWS CLI v2-y
RUN yum install sudo java-17-amazon-corretto unzip tar -y
```

```
RUN sudo yum remove awscli -y
RUN curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
 "awscliv2.zip"
RUN sudo unzip awscliv2.zip
RUN sudo ./aws/install

#·Installation dir
RUN mkdir -p /usr/local/velocity/installation/gapwalk

# Copy PlanetsDemo archive to a dedicated apps dir
COPY PlanetsDemo-v1.zip /workdir/apps/

# Copy resources (tomcat, blu age runtime) to installation dir
COPY tomcat.tar.gz /usr/local/velocity/installation/tomcat.tar.gz
COPY aws-bluage-runtime-4.x.x.tar.gz /usr/local/velocity/installation/gapwalk/
gapwalk.tar.gz

# run relevant installation scripts
RUN ./install-gapwalk.sh
RUN ./install-app.sh

EXPOSE 8080
EXPOSE 8081
# ...

WORKDIR /bluage/tomcat.gapwalk/velocity
# Run Command to start Tomcat server
CMD ["sh", "-c", "sudo bin/catalina.sh run"]
```

以下是的內容`install-gapwalk.sh`。

```
#!/bin/sh

# Vars
TEMP_DIR=/bluage/tomcat.gapwalk/temp

# Install
echo "Installing Gapwalk and Tomcat"
sudo rm -rf /bluage
mkdir -p ${TEMP_DIR}
# Copy Blu Age runtime and tomcat archives to temporary extraction dir
sudo cp /usr/local/velocity/installation/gapwalk/gapwalk-bluage.tar.gz ${TEMP_DIR}
sudo cp /usr/local/velocity/installation/tomcat.tar.gz ${TEMP_DIR}
```

```
# Create velocity dir
mkdir -p /bluage/tomcat.gapwalk/velocity
# Extract tomcat files
tar -xvf ${TEMP_DIR}/tomcat.tar.gz -C ${TEMP_DIR}
# Copy all tomcat files to velocity dir
cp -fr ${TEMP_DIR}/apache-tomcat-10.x.x/* /bluage/tomcat.gapwalk/velocity
# Remove default webapps of Tomcat
rm -f /bluage/tomcat.gapwalk/velocity/webapps/*
# Extract Blu Age runtime at velocity dir
tar -xvf ${TEMP_DIR}/gapwalk-bluage.tar.gz -C /bluage/tomcat.gapwalk
# Remove temporary extraction dir
sudo rm -rf ${TEMP_DIR}
```

以下是的內容`install-app.sh`。

```
#!/bin/sh

APP_DIR=/workdir/apps
TOMCAT_GAPWALK_DIR=/bluage/tomcat.gapwalk

unzip ${APP_DIR}/PlanetsDemo-v1.zip -d ${APP_DIR}
cp -r ${APP_DIR}/webapps/* ${TOMCAT_GAPWALK_DIR}/velocity/webapps/
cp -r ${APP_DIR}/config/* ${TOMCAT_GAPWALK_DIR}/velocity/config/
```

2. 提供您在`application-main.yml`檔案中下列程式碼片段 (位於資料`{TOMCAT_GAPWALK_DIR}/config`夾中) 中作為先決條件之一部分所建立之資料庫的連線資訊。如需詳細資訊，請參閱建立並連線至 Aurora Postgre 資SQL料庫叢集。

```
datasource:
   jicsDs:
     driver-class-name :
     url:
     username:
     password:
     type :
```

3. 建立映像檔並將其推送至您的 Amazon ECR 儲存庫。如需指示，請參閱 Amazon 彈性容器登錄使用者指南中的推送 Docker 映像。然後，根據您的情況，使用 Amazon ECR 映像建立 Amazon EKS 網繭或 Amazon ECS 任務定義，並將其部署到叢集。例如，請參閱 Amazon 彈性容器服務開發人員指南中的使用主控台建立任務定義和 Amazon 使用EKS者指南中的部署範例應用程式。

測試部署的應用程式

如需如何測試 PlanetsDemo 應用程式的範例，請參閱the section called “測試 PlanetsDemo 應用程式”。

升級 Amazon 上的 AWS 藍光時代運行時 EC2

本指南介紹瞭如何在 Amazon 上升級 AWS 藍光時代運行時EC2。

主題

- 必要條件
- 升級 Amazon EC2 實例中的 AWS 藍光時代運行時
- 升級容器中的 AWS 藍光時代運行時

必要條件

開始之前，請確定您符合下列先決條件。

- 若要檢查您的版本是否有特定指示，請參閱the section called “升級 AWS 藍光時代”。
- 完成the section called “AWS 藍色年齡運行時先決”和the section called “入職 AWS 藍光時代運行時”。
- 確保您擁有一個包含最新的 AWS 藍光時代運行時的 Amazon EC2 實例。如需詳細資訊，請參閱開始使用 Amazon EC2 Linux 執行個體。
- 確保您可以成功連接到 Amazon EC2 執行個體，例如使用SSM.
- 下載您要升級到的 AWS 藍光時代運行時的版本。有關更多信息，請參閱the section called “ 設置 AWS 藍光時代運行時（ 非託管 ）”框架由兩個二進製文件組成：`aws-bluage-runtime-x.x.x.x.tar.gz`和`aws-bluage-webapps-x.x.x.x.tar.gz`。

升級 Amazon EC2 實例中的 AWS 藍光時代運行時

請完成以下步驟來升級 AWS 藍光時代執行階段。

1. Connect 到您的 Amazon EC2 實例，並通過運行以下命令將用戶更改為 su。

   ```
   sudo su
   ```

   在本教學課程中，您需要超級使用者權限才能執行指令。

2. 建立兩個資料夾，每個二進位檔案各一個。
3. 使用與二進位檔案相同的名稱命名每個資料夾。
4. 將每個二進製文件複製到相應的文件夾。

> **Warning**
>
> 提取每個二進製文件會生成具有相同名稱的文件夾。因此，如果您一個接一個地在相同位置解壓縮兩個二進位檔案，則會覆寫內容。

5. 若要擷取二進位檔案，請使用下列命令。運行每個文件夾中的命令。

```
tar xvf aws-bluage-runtime-x.x.x.x.tar.gz
tar xvf aws-bluage-webapps-x.x.x.x.tar.gz
```

6. 通過使用以下命令停止阿帕奇 Tomcat 服務。

```
systemctl stop tomcat.service
systemctl stop tomcat-webapps.service
```

7. <your-tomcat-path>/shared/以的內容取代的內容aws-bluage-runtime-x.x.x.x/velocity/shared/。
8. 使用 aws-bluage-runtime-x.x.x.x/velocity/webapps/gapwalk-application.war 取代 <your-tomcat-path>/webapps/gapwalk-application.war。
9. 將中<your-tomcat-path>/webapps/的 war 檔案取代為相同的檔案aws-bluage-webapps-x.x.x.x/velocity/webapps/。bac.war jac.war
10. 通過運行以下命令啟動阿帕奇 Tomcat 服務。

```
systemctl start tomcat.service
systemctl start tomcat-webapps.service
```

11. 檢查日誌。

若要檢查已部署應用程式的狀態，請執行下列命令。

```
curl http://localhost:8080/gapwalk-application/
```

應該會顯示以下訊息。

```
Jics application is running
```

```
curl http://localhost:8181/jac/api/services/rest/jicsservice/
```

應該會顯示以下訊息。

```
Jics application is running
```

```
curl http://localhost:8181/bac/api/services/rest/bluesamserver/serverIsUp
```

響應應為空。

AWS 藍光時代執行階段已成功升級。

升級容器中的 AWS 藍光時代運行時

請完成以下步驟來升級 AWS 藍光時代執行階段。

1. 使用所需的 AWS 藍光時代運行時版本重建 Docker 映像。如需說明，請參閱 the section called “在 Amazon 上設置 AWS 藍光時代運行時（非託管）EC2”。
2. 將您的 Docker 映像推送到您的 Amazon ECR 存儲庫。
3. 停止並重新啟動 Amazon ECS 或 Amazon EKS 服務。
4. 檢查日誌。

AWS 藍光時代運行時已成功升級。

設置 AWS 藍光時代運行時（在 Amazon 上EC2）Amazon CloudWatch 警報

您可以設定 CloudWatch 接收應用程式記錄檔，並新增警示來警告您可能發生的錯誤。這可讓您在部署的應用程式遇到例外狀況時收到更明顯的通知。以下各節可協助您瞭解並瞭解記 CloudWatch 錄和警示設定的組態。

部署記 CloudWatch 錄

依預設，`application-main.yml`檔案會包含另一個名為的記錄設定檔的參考`logback-cloudwatch.yml`。

```
logging:
  config: classpath:logback-cloudwatch.xml
```

這兩個文件都在 config 文件夾中，這是如何配置 CloudWatch 日誌記錄，如以下各節所述。

CloudWatch 日誌記錄的配置

預設`logback-cloudwatch.xml`檔案具有下列內容。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration>
<configuration>

    <appender name="console" class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <pattern>%date{yyyy-MM-dd HH:mm:ss.SSS,UTC}  %level --- [%thread{15}]
 %logger{40} : %msg%n%xThrowable</pattern>
        </encoder>
    </appender>

    <appender name="cloudwatch"
 class="com.netfective.bluage.runtime.cloudwatchlogger.CloudWatchAppender">
        <logGroup>BluAgeRuntimeOnEC2-Logs</logGroup>
        <logStream>%date{yyyy-MM-dd,UTC}.%instanceId.%uuid</logStream>
        <layout>
            <pattern>%date{yyyy-MM-dd HH:mm:ss.SSS,UTC}  %level --- [%thread{15}]
 %logger{40} : %msg%n%xThrowable</pattern>
        </layout>
        <appender-ref ref="console" />
    </appender>

    <root level="INFO">
        <appender-ref ref="cloudwatch" />
    </root>
</configuration>
```

<appender name="cloudwatch"/>元素之外的所有內容都是標準的記錄配置。有這個文件中的兩個追加程序：控制台追加程序日誌發送到控制台和 CloudWatch 追加程序發送日誌。 CloudWatch

`root`元素中的`level`屬性指定整個應用程式的記錄層級。

標籤內的必要值<appender name="cloudwatch"/>是：

- < logGroup />：設定中記錄群組的名稱。 CloudWatch如果未指定該值，則默認為`BluAgeRuntimeOnEC2-Logs`。如果記錄群組不存在，則會自動建立該記錄群組。這種行為可以通過配置進行更改，如下所述。
- < logStream />：設定中 logStream (記錄群組內部) 的名稱。 CloudWatch

可選值：

- <region/>：覆寫記錄串流將寫入的區域。根據預設，記錄會移至與EC2執行個體相同的區域。
- <layout/>：記錄訊息將使用的模式。
- <maxbatchsize/>： CloudWatch 每個操作要發送的最大日誌消息數。
- <maxbatchtimemillis/>：允許寫入 CloudWatch 記錄的時間 (以毫秒為單位)。
- <maxqueuewaittimemillis/>：嘗試在內部記錄佇列中插入要求的時間 (以毫秒為單位)。
- <internalqueuesize/>：內部佇列的大小上限。
- <createlogdests/>：如果記錄群組不存在，請建立記錄群組和記錄串流。
- <initialwaittimemillis/>：您希望線程在啟動時睡眠的時間量。此初始等待允許初始累積記錄。
- <maxeventmessagesize/>：記錄事件的大小上限。超過此大小的記錄將不會傳送。
- <truncateeventmessages/>：截斷太長的郵件。
- <printrejectedevents/>：啟用緊急附加程序。

CloudWatch 設置

為了使上述組態能夠正確地將日誌推送到 CloudWatch，請更新 Amazon EC2 IAM 執行個體設定檔角色，以授與 `BluAgeRuntimeOnEC2-Logs` 日誌群組及其日誌串流的其他許可：

- `logs:CreateLogStream`
- `logs:DescribeLogStreams`
- `logs:CreateLogGroup`
- `logs:PutLogEvents`
- `logs:DescribeLogGroups`

警報設定

借助 CloudWatch 日誌，您可以根據應用程序和需求配置不同的指標和警報。具體而言，您可以為使用情況警示設定主動警示，以便在可能使您的應用程式處於寬限期的錯誤時收到警告 (最後，完全防

止應用程式運作)。為此，您可以在日誌中添加有關「錯誤 C5001」字符串的指標，該指標突出顯示與 AWS Blu Age 控制系統連接的錯誤。然後，您可以定義對此量度做出反應的警示。

在 Amazon 上的 AWS 藍光時代運行時中設置許可依賴項 EC2

本指南介紹瞭如何設置可與 Amazon 上的 AWS Blu Age 運行時一起使用的其他許可依賴關係EC2。

主題

- 必要條件
- 概觀
- 設置JAC和 BAC Web 應用程序的依賴關係

必要條件

開始之前，請確定您已完成下列先決條件。

- 完成the section called “AWS 藍色年齡運行時先決”和the section called “入職 AWS 藍光時代運行時”。
- 確保您有一個包含最新的 AWS 藍光時代運行時的 Amazon EC2 實例（在 Amazon 上EC2）。如需詳細資訊，請參閱開始使用 Amazon EC2 Linux 執行個體。
- 確保您可以成功連接到 Amazon EC2 執行個體，例如使用SSM.
- 從其來源取得下列相依性。

Oracle 資料庫

提供 Oracle 資料庫驅動程式。我們使用 ojdbc11-23.0.23.09. jar 版本測試了 AWS 藍光時代運行時（在 Amazon 上EC2）功能，但是更新的版本可能兼容。

IBMMQ 連接

提供 IBMMQ 用戶端。我們使用 .ibm.mq.jakarta.client 9.3.4.1.jar 版本測試了 AWS 藍光時代運行時（在 Amazon 上EC2）功能，但是更新的版本可能兼容。

使用此依賴版本，還提供以下傳遞依賴關係：

- 瓶裝百分之十五至十七
- BCPKIX-罐裝

- 布庫蒂爾-日本料十五至十七

DDS印表機檔案

提供賈斯珀報告庫。我們使用 jasperreports-6.16.0.jar 測試了 AWS 藍光時代運行時（在 Amazon 上 EC2）功能，但是更新的版本可能兼容。

使用此依賴版本，還提供以下傳遞依賴關係：

- 田園核心 -1.4.1.jar
- 罐子-XML-1.4.1.罐
- 普通消化器 -2.1.1 罐
- 心形紅酒罐
- 項目 -2.1.7.J8.jar
- 噴射器 1. 罐
- 日本料理 -1.0.23 罐
- 自由圖 -1.0.19 罐
- 共同的無簷小豆 -1.9.4.jar
- 公用集合 -3.2.2.jar

概觀

若要安裝相依性，請完成以下步驟。

1. Connect 到您的 Amazon EC2 實例，並通過運行以下命令將用戶更改為 su。

   ```
   sudo su
   ```

   在本教學課程中，您需要超級使用者權限才能執行指令。
2. 導覽至 <your-tomcat-path>/extra/ 資料夾。

   ```
   cd <your-tomcat-path>/extra/
   ```
3. 根據需要在此文件夾複製上述任何依賴關係。
4. 通過運行以下命令停止並啟動 tomcat.service。

```
systemctl stop tomcat.service
```

```
systemctl start tomcat.service
```

5. 檢查服務的狀態，確定服務正在執行。

```
systemctl status tomcat.service
```

6. 驗證記錄檔。

設置JAC和 BAC Web 應用程序的依賴關係

1. 如果您的JICS或 Blusam 數據庫託管在 Oracle 上，那么您需要在中提供 Oracle 數据庫驅動程序。`<your-tomcat-path>/extra`
2. 如果資料夾不存在，請建立該資料夾。
3. 停止並重新啟動您的阿帕奇 Tomcat 服務器。
4. 驗證記錄檔。

測試 PlanetsDemo 應用程式

若要檢查已部署 PlanetsDemo 應用程式的狀態，請在取代後執行下列命令 `load-balancer-DNS-namelistener-port`，並`web-binary-name`使用正確的設定值執行下列命令。

```
curl http://load-balancer-DNS-name:listener-port/gapwalk-application/
```

如果應用程式正在執行，您會看到下列輸出訊息：`Jics application is running`。

接下來，運行以下命令。

```
curl http://load-balancer-DNS-name:listener-port/jac/api/services/rest/jicsservice/
```

如果應用程式正在執行，您會看到下列輸出訊息：`Jics application is running`。

```
Jics application is running
```

如果您已配置 Blusam，則在運行以下命令時可能會出現空白響應。

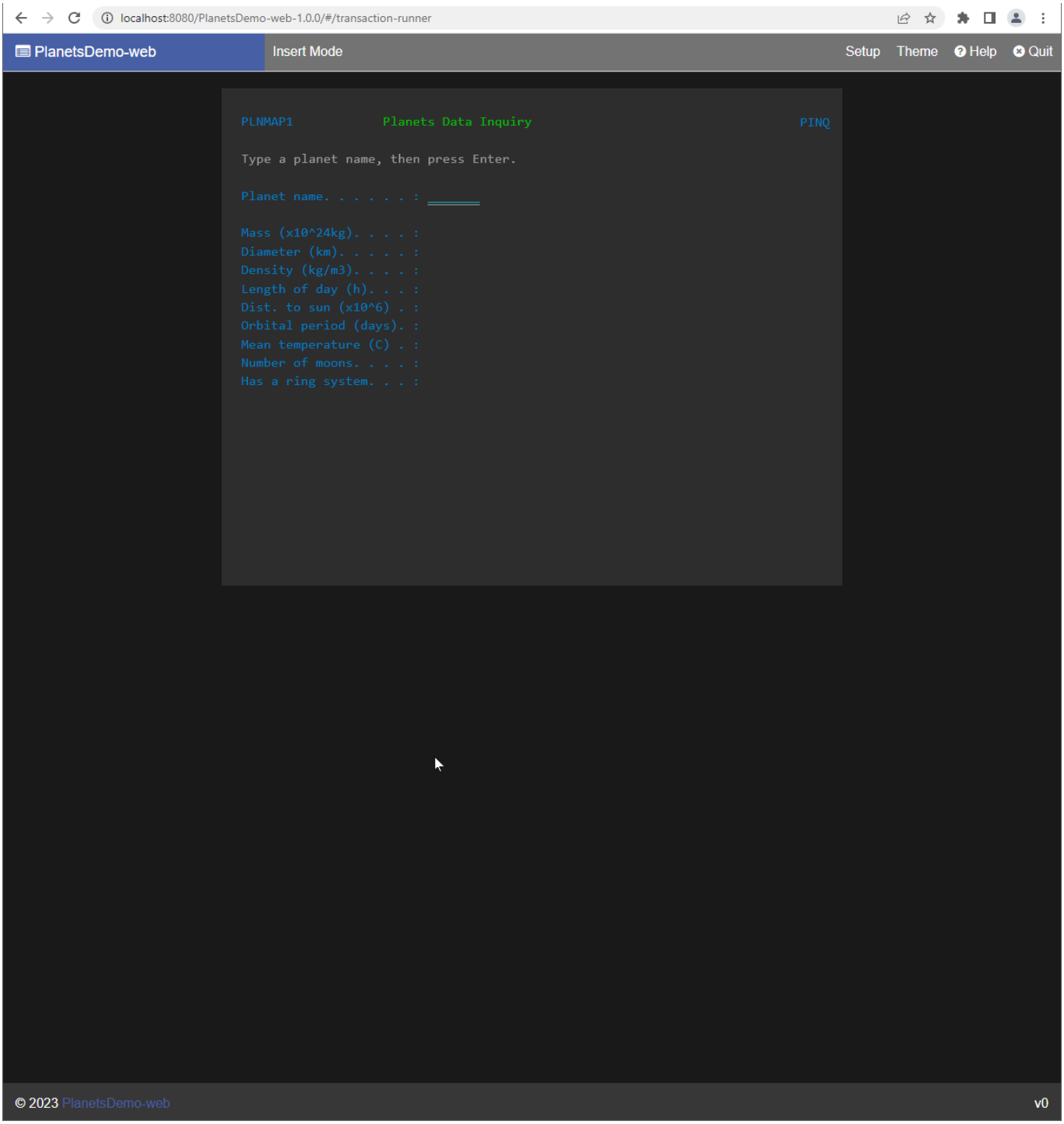
```
curl http://load-balancer-DNS-name:listener-port/bac/api/services/rest/bluesamserver/
serverIsUp
```

請注意網絡二進製文件的名稱（PlanetsDemo-web-1.0.0，如果不變）。若要存取 PlanetsDemo 應用程式，請使用下列格式URL之一。

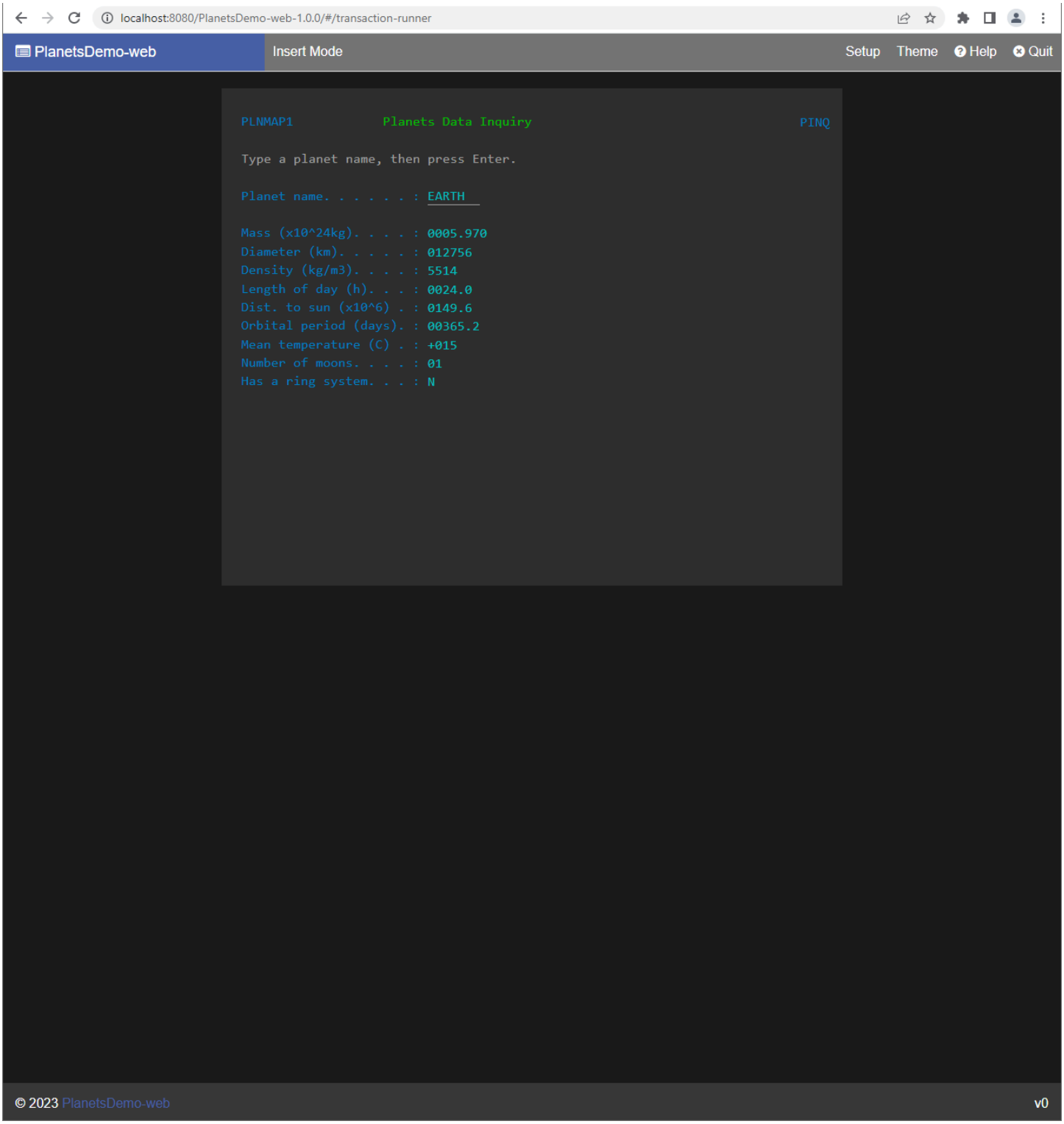
```
https://load-balancer-DNS-name:listener-port/web-binary-name
```

PlanetsDemo 應用程式啟動後，會顯示首頁。

PINQ在文字方塊中輸入，然後按 Enter 鍵。顯示資料查詢頁面。



例如，EARTH在 PlanetsDemo 名稱欄位中輸入，然後按 Enter。顯示您輸入的行星頁面。



使用藍光時代開發人員修改源代碼 IDE

如果您使用的是 AWS-managed AWS Blu Age 運行時引擎，則可以使用 Blu Age 開發人員修改生成的源代碼。如果出於某種原因需要更新現代化代碼，或者某部分舊版源代碼無法現代化，則可能需要

執行此操作。您訪問藍光時代開發人員通過 Amazon AppStream 2.0。本節介紹如何在 AppStream 2.0 上設置藍光時代開發人員。它還說明瞭如何使用 Blu Age 開發者更新源代碼，使用示例應用程序 PlanetsDemo。

主題

- 教程：為 AWS 藍光時代開發人員設置 AppStream 2.0 IDE
- 教程：在 AppStream 2.0 上使用 AWS 藍光時代開發人員

教程：為 AWS 藍光時代開發人員設置 AppStream 2.0 IDE

AWS 大型主機現代化透過 Amazon 2.0 提供多種工具。 AppStream AppStream 2.0 是全受管、安全的應用程式串流服務，可讓您將桌面應用程式串流給使用者，而無需重新撰寫應用程式。 AppStream 2.0 為使用者提供即時存取所需的應用程式，並在他們選擇的裝置上提供反應靈敏、流暢的使用者體驗。使用 AppStream 2.0 託管執行階段引擎特定工具，可讓客戶應用程式團隊直接從 Web 瀏覽器使用這些工具，與存放在 Amazon S3 儲存貯體或 CodeCommit儲存庫中的應用程式檔案互動。

如需 2.0 版瀏覽器 Support 的相關資訊，請參閱《Amazon AppStream AppStream 2.0 管理指南》中的系統需求和功能支援 (網頁瀏覽器)。如果您在使用 AppStream 2.0 時遇到問題，請參閱《Amazon AppStream 2.0 管理指南》中的 AppStream 2.0 使用者問題疑難排解。

本文件說明如何在 AppStream 2.0 車隊IDE上設定 AWS 藍光時代開發人員。

主題

- 先決條件
- 步驟 1：建立 Amazon S3 儲存貯體
- 步驟 2：將政策附加到 S3 儲存貯體
- 步驟 3：將檔案上傳到 Amazon S3 儲存貯體
- 步驟 4：下載 AWS CloudFormation 範本
- 步驟 5：使用以下方式建立車隊 AWS CloudFormation
- 步驟 6：存取執行個體
- 清除資源

先決條件

下載包含IDE在 AppStream 2.0 下設置 AWS Blu Age 開發人員所需的工件的存檔文件。

> **Note**
>
> 這是一個大文件。如果您在操作逾時時遇到問題，建議您使用 Amazon EC2 執行個體來改善上傳和下載效能。

步驟 1：建立 Amazon S3 儲存貯體

在與您要建立的 AppStream 2.0 叢集相 AWS 區域 同的位置建立 Amazon S3 儲存貯體。此值區將包含您完成本教學課程所需的成品。

步驟 2：將政策附加到 S3 儲存貯體

將下列原則附加至您為本教學課程建立的值區。請務必以您建立MYBUCKET的值區的實際名稱取代。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "AllowAppStream2.0ToRetrieveObjects",
        "Effect": "Allow",
        "Principal": {
            "Service": "appstream.amazonaws.com"
        },
        "Action": "s3:GetObject",
        "Resource": "arn:aws:s3:::MYBUCKET/*"
    }]
}
```

步驟 3：將檔案上傳到 Amazon S3 儲存貯體

解壓縮您在先決條件中下載的檔案，然後將appstream資料夾上傳到值區。上傳此資料夾會在值區中建立正確的結構。如需詳細資訊，請參閱 Amazon S3 使用者指南中的上傳物件。

步驟 4：下載 AWS CloudFormation 範本

下載以下 AWS CloudFormation 範本。您需要這些範本來建立和填入 AppStream 2.0 叢集。

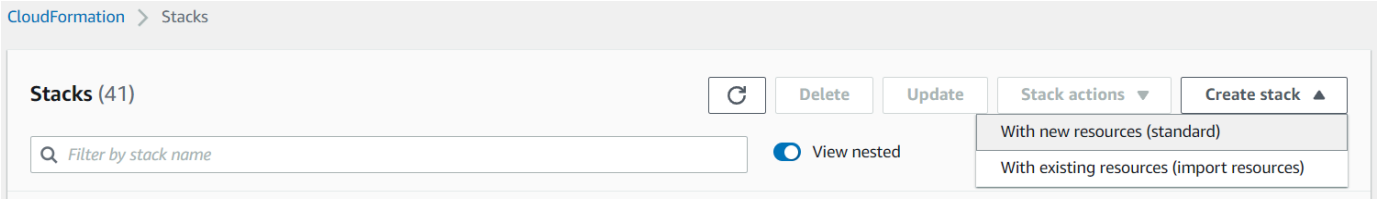
- CFN-平方米-. 羊 appstream-elastic-fleet-linux
- CFN-平方米-亞麻油 appstream-bluage-dev-tools
- CFN-平方米-. 羊 appstream-bluage-shared-linux
- CFN-平方米-. 羊 appstream-chrome-linux

- CFN-平方米-. 羊 appstream-eclipse-jee-linux
- CFN-平方米-. 羊 appstream-pgadmin-linux

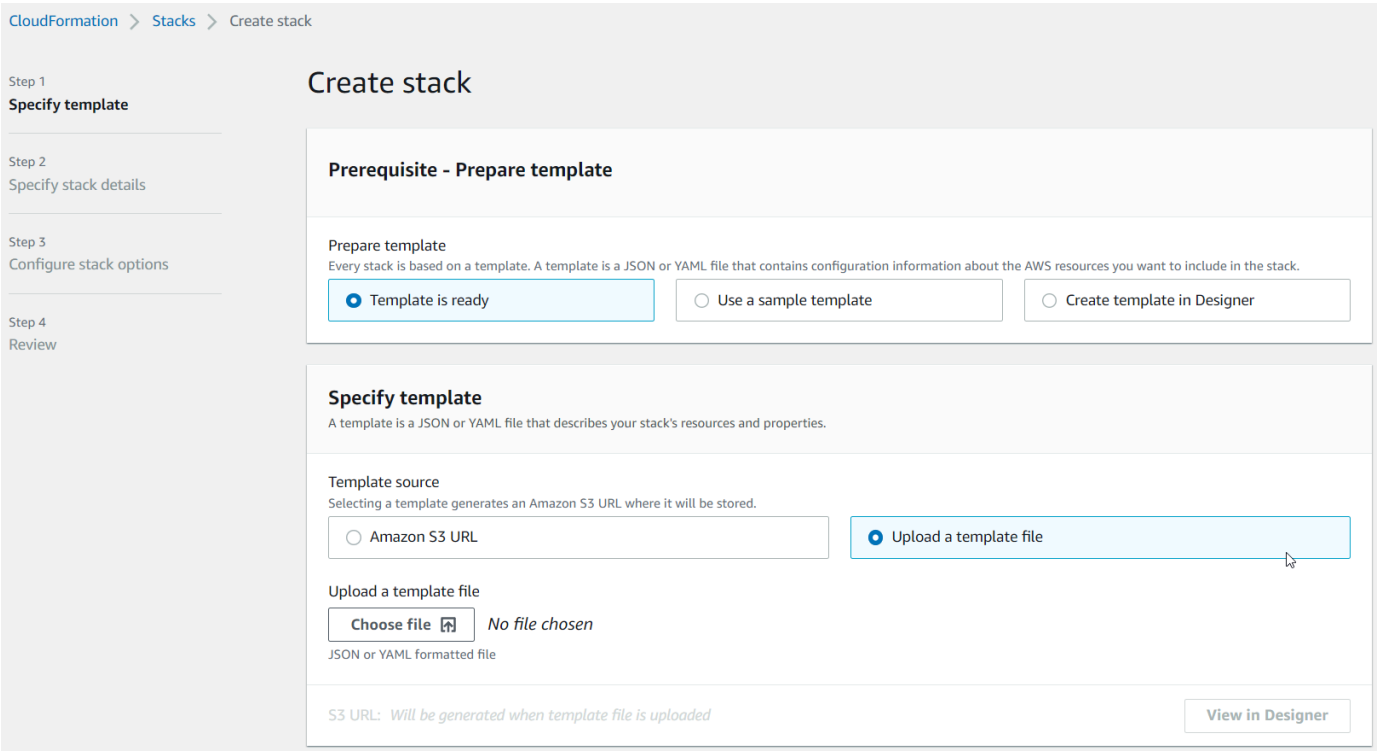
步驟 5：使用以下方式建立車隊 AWS CloudFormation

在此步驟中，您可以使用`cfn-m2-appstream-elastic-fleet-linux.yaml` AWS CloudFormation 範本建立 AppStream 2.0 叢集和堆疊來託管 AWS Blu Age 開發人員IDE。建立叢集和堆疊之後，您將執行在上一個步驟中下載的其他 AWS CloudFormation 範本，以安裝開發人員IDE和其他必要工具。

1. 在「 AWS 管理」主控台 AWS CloudFormation 中導覽至，然後選擇「堆疊」。
2. 在 [堆疊] 中，選擇 [建立堆疊] 和 [使用新資源 (標準)]：



3. 在 [建立堆疊] 中，選擇 [範本已就緒] 並 [上傳範本檔案]：



4. 選擇「選擇檔案」，然後瀏覽至檔案`cfn-m2-appstream-elastic-fleet-linux.yaml`。選擇 Next (下一步)。

5. 在指定堆疊詳細資料中，提供下列資訊：

   - 堆疊的名稱。
   - 您的預設安全性群組和該安全性群組的兩個子網路。

   > **Note**
   >
   > 安全性群組的兩個子網路必須位於不同的可用區域。

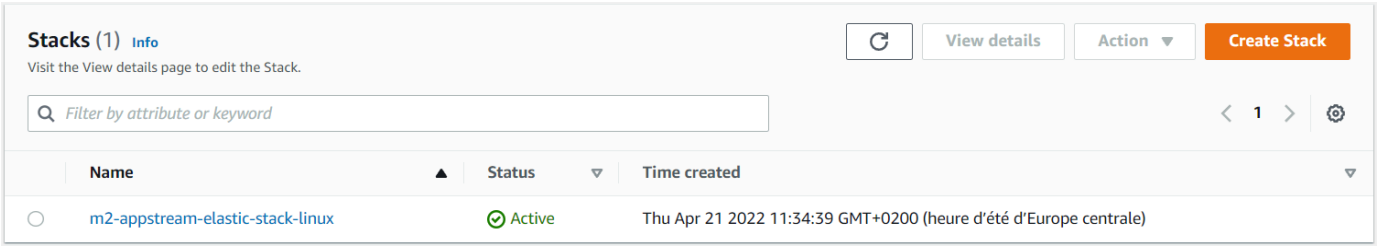
6. 選擇 [下一步]，然後再選擇 [下一步]。
7. 選擇 [我確認 AWS CloudFormation 可能會使用自訂名稱建立IAM資源]。 ，然後選擇「提交」。
8. 建立叢集之後，請使用其他下載的範本建立 CloudFormation 堆疊，以完成應用程式的設定。請務必BucketName每次更新以指向正確的 S3 儲存貯體。您可以在 CloudFormation主控台BucketName中編輯。或者，您可以直接編輯範本檔案並更新S3Bucket性質。

> **Note**
>
> 下載的範本預期會在資料夾結構名為的 S3 儲存貯體中尋找資產`appstream/bluage/developer-ide/`。值區必須與您建立的叢集位於 AWS 區域 相同的位置。

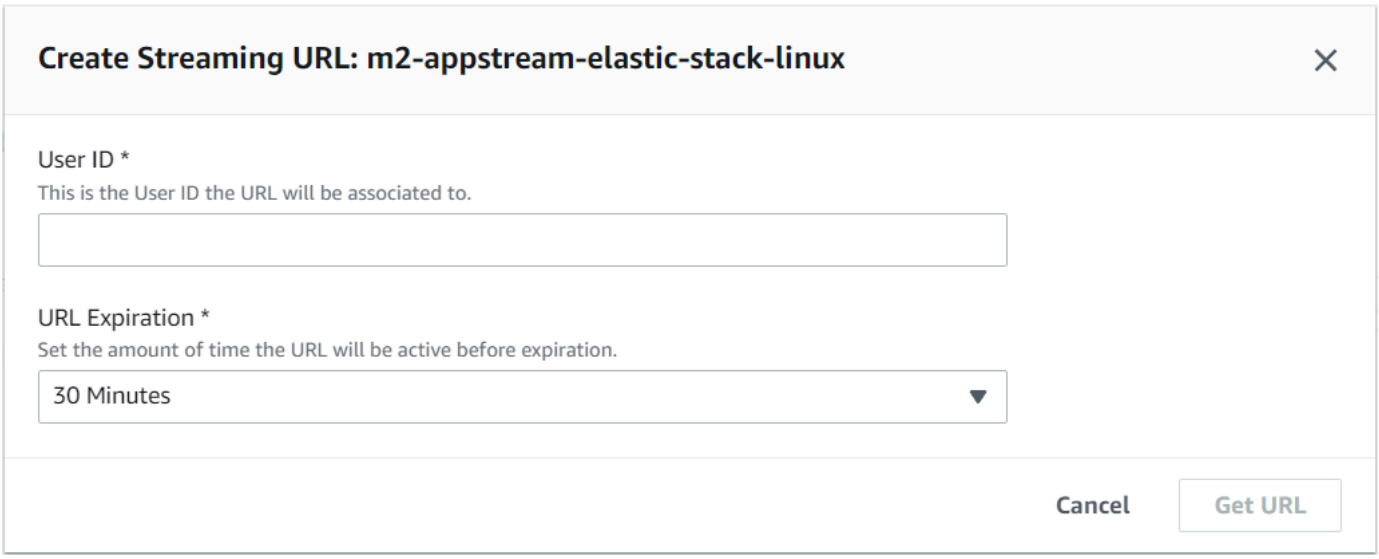
步驟 6：存取執行個體

建立並啟動叢集之後，您可以建立暫時連結，透過原生用戶端存取叢集。

1. 導航到 AppStream 2.0， AWS Management Console 然後選擇先前創建的堆棧：



2. 在堆疊詳細資料頁面上，選擇「動作」，然後選擇「建立串流」URL：



3. 在 [建立串流] 中URL，輸入任意使用者 ID 和URL到期時間，然後選擇 [取得] URL。您可以使用URL它來流式傳輸到瀏覽器或本機客戶端。我們建議您串流至原生用戶端。

清除資源

如需清理建立的堆疊和叢集的程序，請參閱建立 AppStream 2.0 叢集和堆疊。

刪除 AppStream 2.0 物件後，您或帳戶管理員也可以清理應用程式設定和主資料夾的 S3 儲存貯體。

> **Note**
>
> 特定使用者的主資料夾在所有叢集中都是唯一的，因此如果同一帳戶中的其他 AppStream 2.0 堆疊處於作用中狀態，您可能需要保留該資料夾。

您無法使用 AppStream 2.0 主控台刪除使用者。相反地，您必須搭配使API用該服務 AWS CLI。如需詳細資訊，請參閱 Amazon AppStream 2.0 管理指南中的使用者集區管理。

教程：在 AppStream 2.0 上使用 AWS 藍光時代開發人員

本教程向您展示如何訪問 AppStream 2.0 上的 AWS Blu Age Developer 並將其與示例應用程序一起使用，以便您可以嘗試使用這些功能。完成本教學課程後，您可以對自己的應用程式使用相同的步驟。

主題

- 步驟 1：建立資料庫
- 步驟 2：存取環境

- 步驟 3：設定執行階段
- 第 4 步：啟動日食 IDE
- 第 5 步：設置一個 Maven 項目
- 步驟 6：設定 Tomcat 伺服器
- 第 7 步：部署到湯姆貓
- 步驟 8：建立資JICS料庫
- 步驟 9：啟動並測試應用程式
- 步驟 10：調試應用程序
- 清除資源

步驟 1：建立資料庫

在此步驟中，您可以使用 Amazon RDS 建立示範應用程式用來儲存組態資訊的受管 Postgre 資SQL料庫。

1. 打開 Amazon RDS 控制台。
2. 選擇資料庫 > 建立資料庫。
3. 選擇 [標準建立] > [下一頁]SQL，保留預設版本，然後選擇 [免費方案]。
4. 選擇資料庫執行個體識別碼。
5. 在「身份證明設定」 中選擇「管理主要認證」 AWS Secrets Manager。如需詳細資訊，請參閱 Amazon RDS 和 Amazon RDS 使用者指南 AWS Secrets Manager中的密碼管理。
6. 請確定VPC與您用於 AppStream 2.0 執行個體的執行個體相同。您可以向管理員詢問此值。
7. 對於VPC安全性群組，請選擇新建。
8. 將「公用」存取權設為 「是」
9. 保留所有其他預設值。檢閱這些值。
10. 選擇建立資料庫。

若要讓資料庫伺服器可從您的執行個體存取，請在 Amazon 中選取資料庫伺服器RDS。在 [連線與安全性] 底下，選擇資料庫伺服器的VPC安全性群組。此安全性群組先前是為您建立的，其說明應該類似於由RDS管理主控台建立的說明。選擇「動作」>「編輯輸入規則」，選擇「新增規則」，然後建立標記類型的規則。SQL對於規則來源，請使用安全性群組預設值您可以開始在「來源」欄位中輸入來源名稱，然後接受建議的識別碼。最後，選擇「儲存規則」。

步驟 2：存取環境

在此步驟中，您可以在 AppStream 2.0 上訪問 AWS 藍光時代開發環境。

1. 請聯絡您的管理員，以取得存取 AppStream 2.0 執行個體的正確方法。如需有關可能用戶端和組態的一般資訊，請參閱《Amazon AppStream 2.0 管理指南》中的 AppStream 2.0 存取方法和用戶端。考慮使用原生用戶端以獲得最佳體驗。
2. 在 AppStream 2.0 中選擇桌面。

步驟 3：設定執行階段

在此步驟中，您將設定 AWS 藍光時代執行階段。您必須在第一次啟動時設定執行階段，如果您收到執行階段升級的通知，請再次設定執行階段 此步驟會填入您的.m2資料夾。

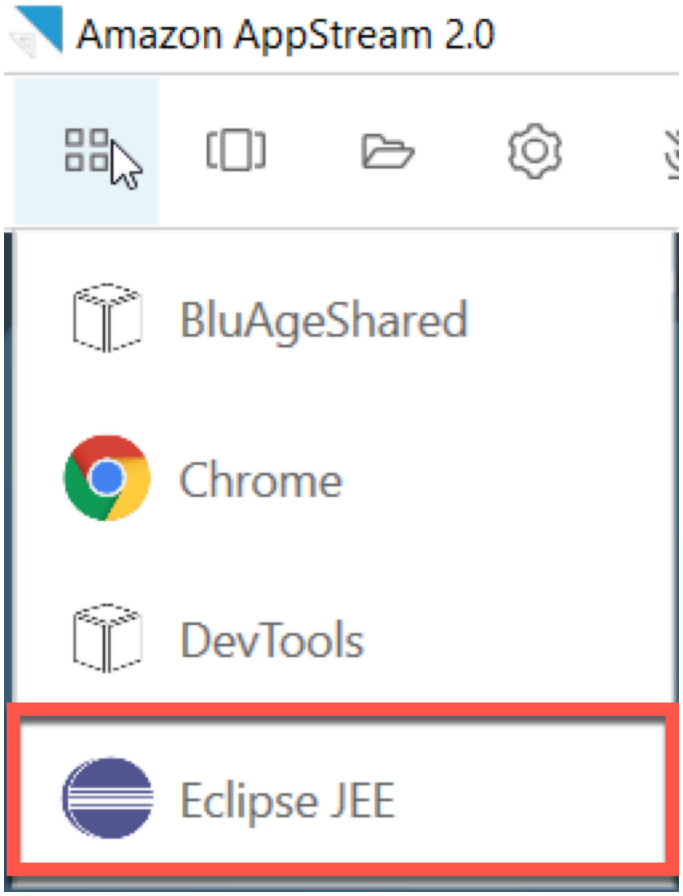
1. 從功能表列選擇 [應用程式]，然後選擇 [終端機]。
2. 輸入以下命令：

```
~/_install-velocity-runtime.sh
```

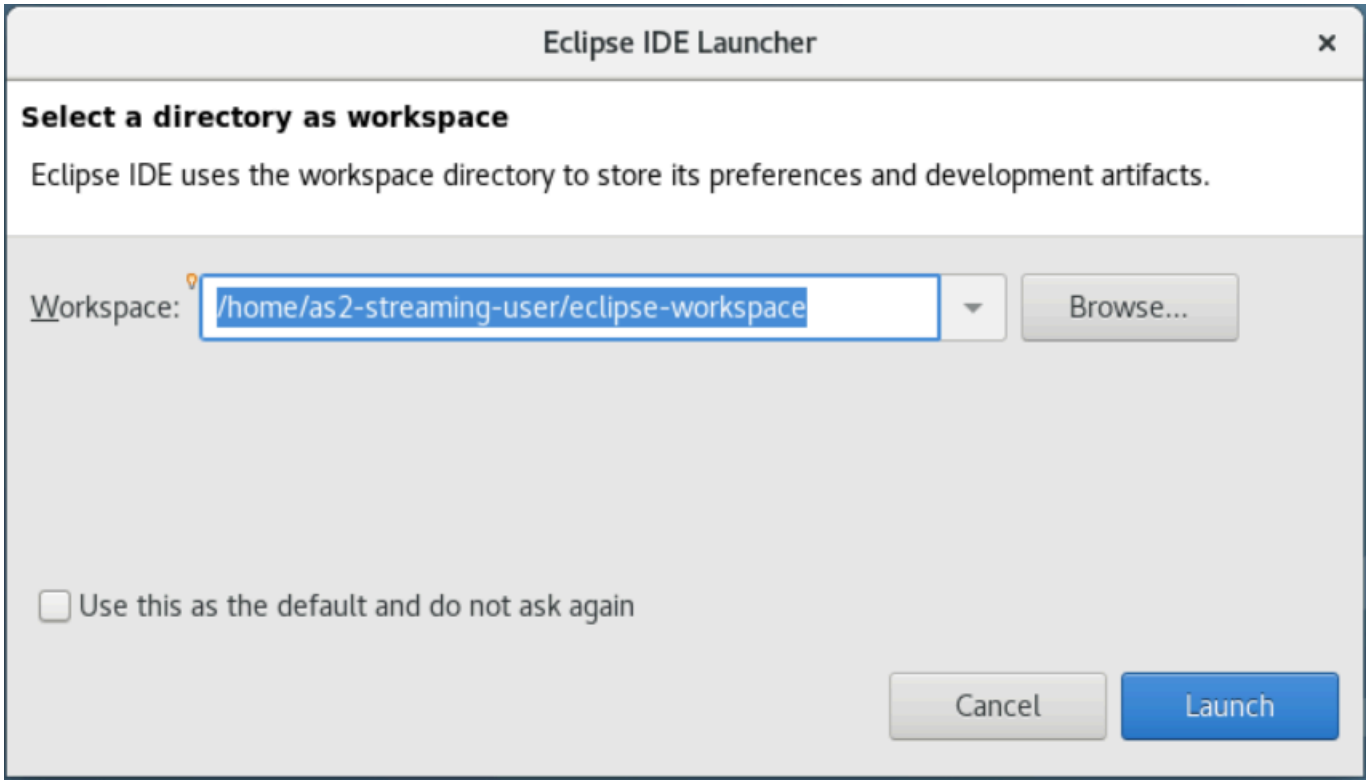
第 4 步：啟動日食 IDE

在此步驟中，啟動 Eclipse IDE 並選擇要在其中創建工作區的位置。

1. 在 AppStream 2.0 中，選擇工具欄上的啟動應用程序圖標，然後選擇 Eclipse JEE。



2. 啟動器開啟後，輸入您要建立工作區的位置，然後選擇 [啟動]。



或者，您可以從命令行啟動 Eclipse，如下所示：

```
~/eclipse &
```

第 5 步：設置一個 Maven 項目

在此步驟中，您會匯入行星示範應用程式的 Maven 專案。

1. 將 PlanetsDemo-pom.zip 上傳至您的個人專屬資料夾。您可以使用本機客戶端「我的文件」功能來執行此操作。
2. 使用指`unzip`令行工具解壓縮檔案。
3. 在解壓縮的文件夾中導航，然後在文本編輯器中打開項目`pom.xml`的根目錄。
4. 編輯`gapwalk.version`屬性，使其符合已安裝的 AWS Blu Age 執行階段。

   如果您不確定安裝的版本，請在終端機中發出以下命令：

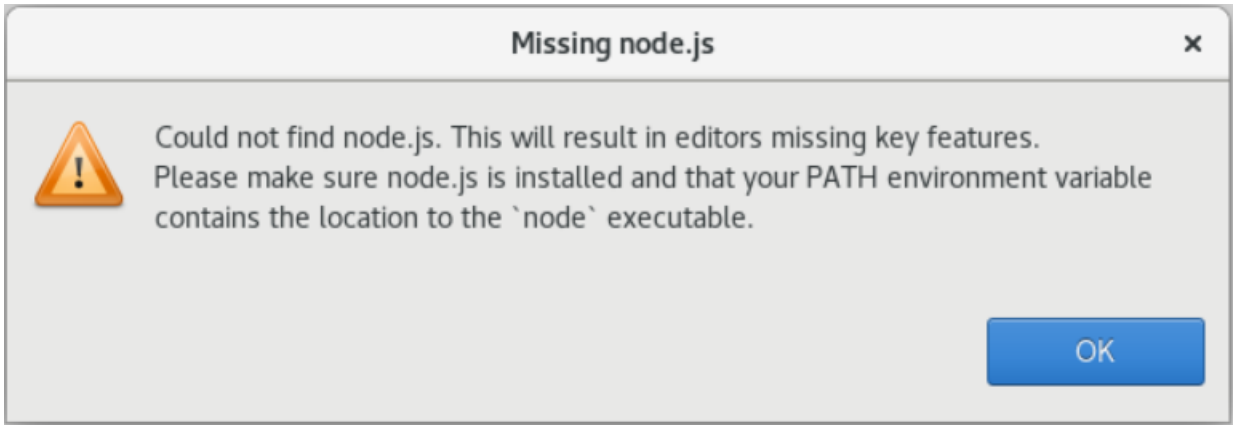
   ```
   cat ~/runtime-version.txt
   ```

   此指令會列印目前可用的執行階段版本，例如，`3.1.0-b3257-dev`。

> **Note**
>
> 請勿在中包含`-dev`尾碼`gapwalk.version`。例如，一個有效的值將是`<gapwalk.version>3.1.0-b3257</gapwalk.version>`。

5. 在 Eclipse 中，選擇檔案，然後選擇匯入。在「匯入」對話方塊視窗中，展開 Maven 並選擇「現有的 Maven 專案」。選擇 Next (下一步)。
6. 在導入 Maven 項目中，提供提取文件的位置，然後選擇完成。

   您可以放心地忽略以下彈出窗口。Maven 下載的本地副本`node.js`來構建項目的角度 (*-網絡) 部分：

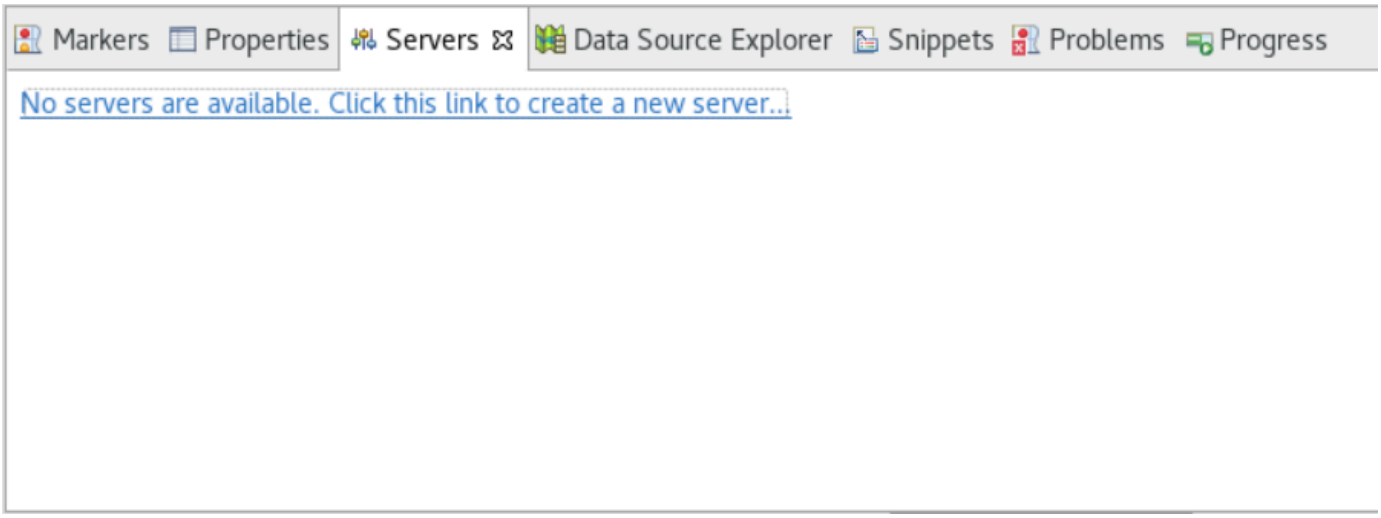
   

   等到構建結束。您可以在「進度」視圖中關注構建。
7. 在 Eclipse 中，選取專案並選擇執行身分。然後選擇安裝。Maven 安裝成功後，它會在下`PlanetsDemoPom/PlanetsDemo-web/target/PlanetsDemo-web-1.0.0.war`面創建`war`文件。

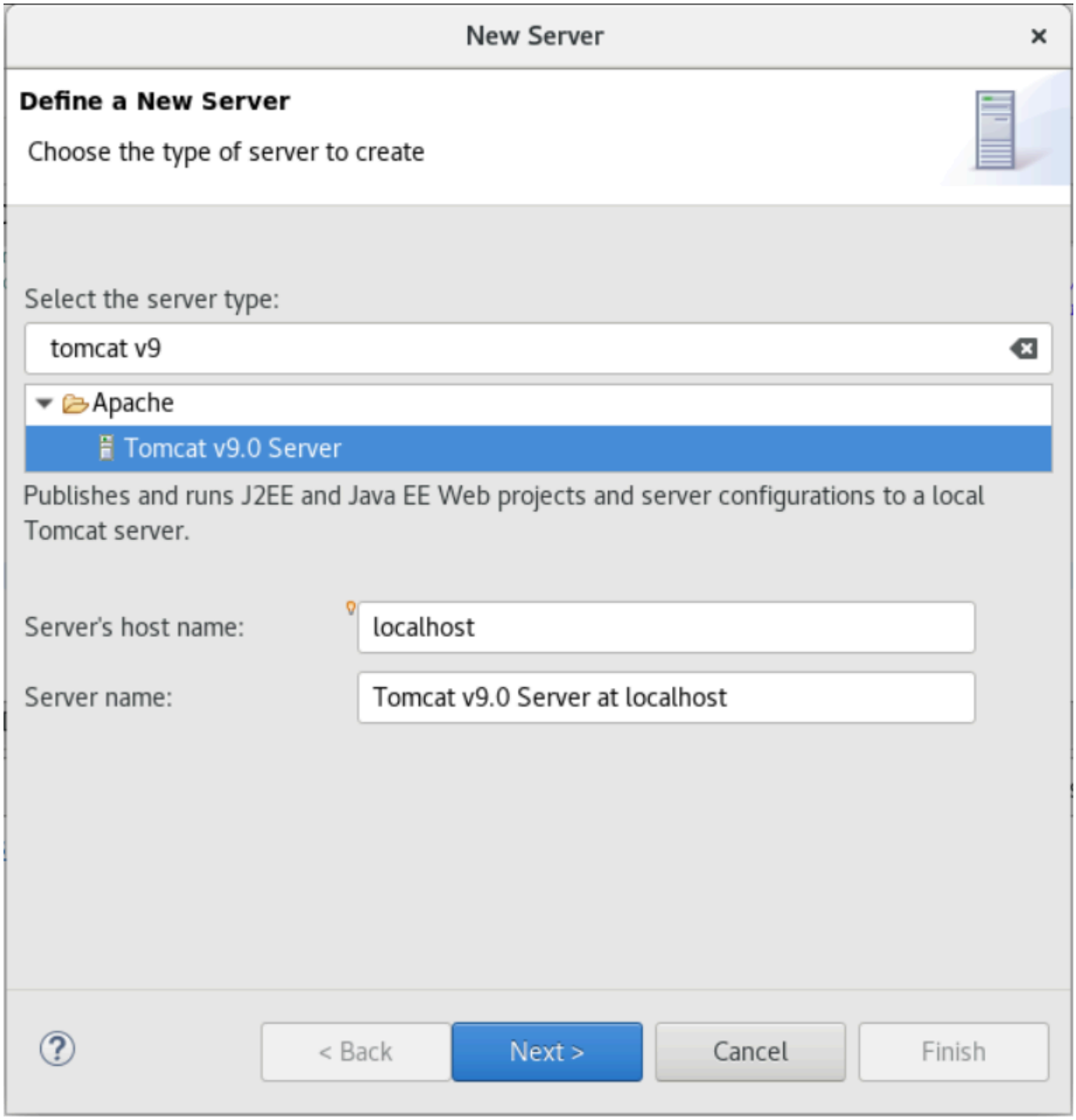
步驟 6：設定 Tomcat 伺服器

在此步驟中，您可以設定 Tomcat 伺服器，您可以在其中部署並啟動已編譯的應用程式。

1. 在 Eclipse 中，選擇「視窗 > 顯示檢視 > 伺服器」以顯示「伺服器」檢視：

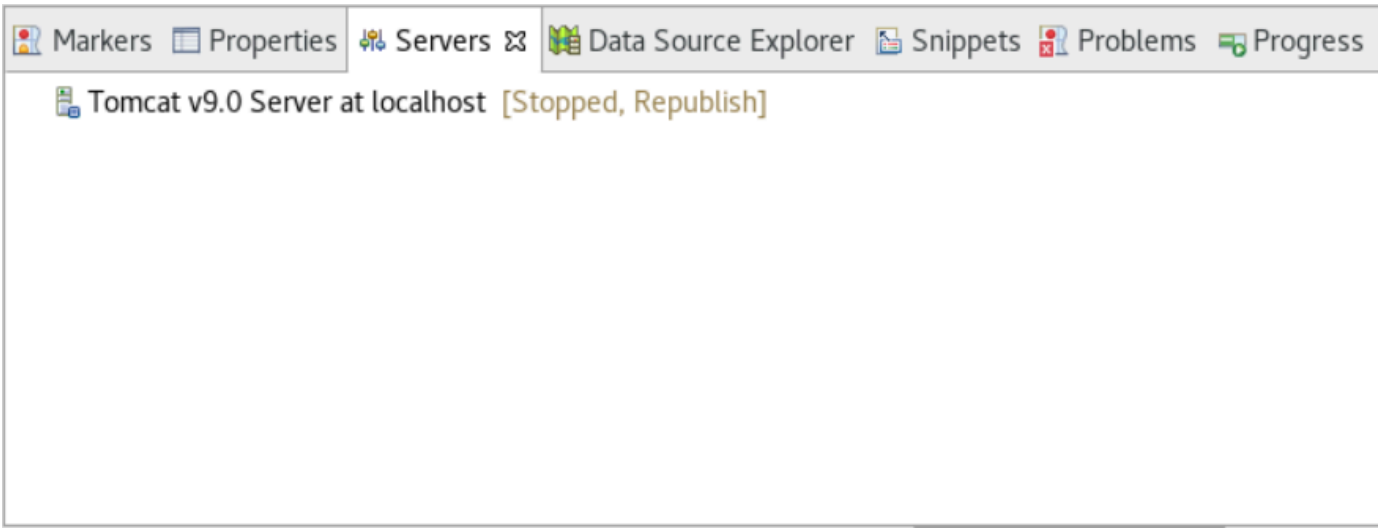


2. 選擇沒有可用的伺服器。點擊此鏈接創建一個新的服務器... 。[新增伺服器] 精靈隨即出現 在精靈的 [選取伺服器類型] 欄位中，輸入 tomcat v9，然後選擇 [T omcat v 9.0 伺服器]。然後選擇下一步。



3. 選擇瀏覽，然後選擇位於主文件夾根目錄的 tomcat 文件夾。保留其預設值JRE，然後選擇「完成」。

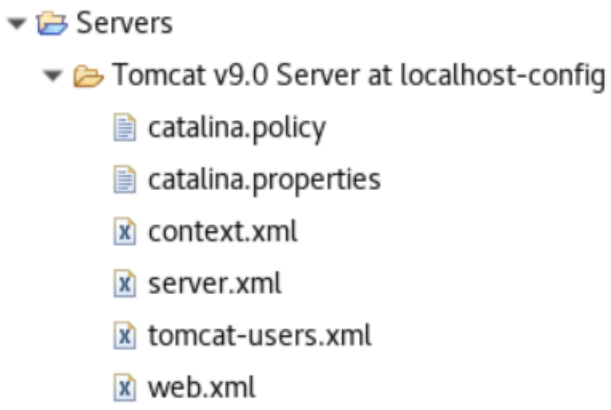
   在工作區中建立「伺服器」專案，而 Tomcat v9.0 伺服器現在可在「伺服器」檢視中使用。這是編譯後的應用程序將被部署和啟動的地方：



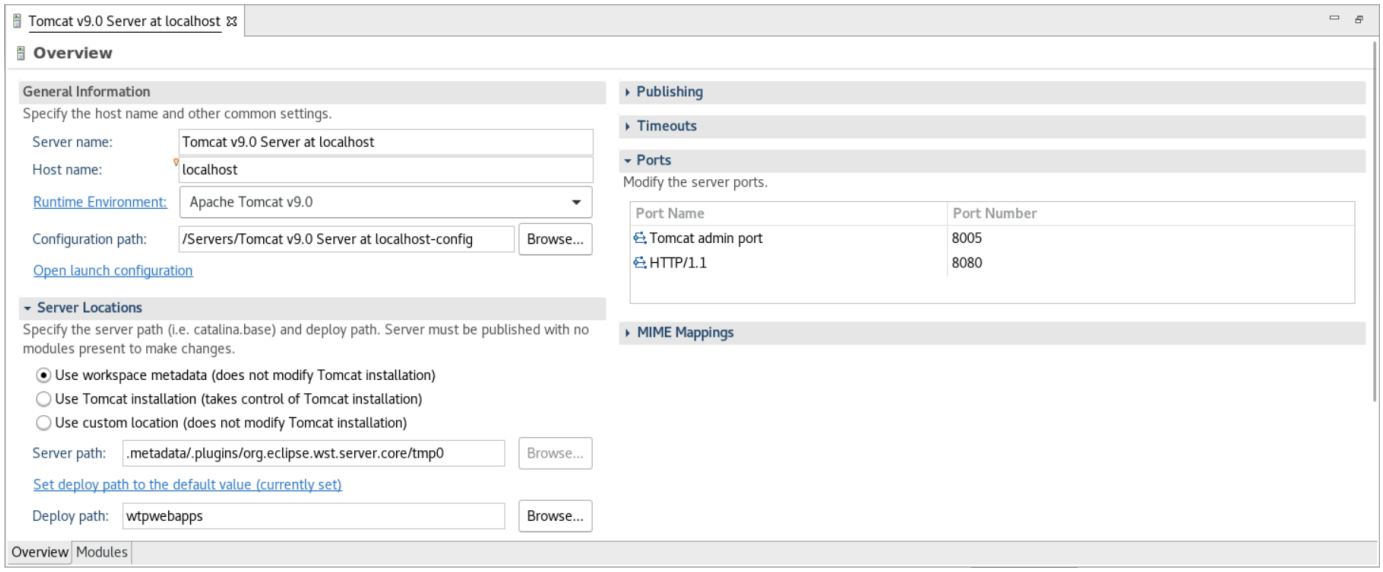
第 7 步：部署到湯姆貓

在此步驟中，您將行星演示應用程序部署到 Tomcat 服務器，以便您可以運行該應用程序。

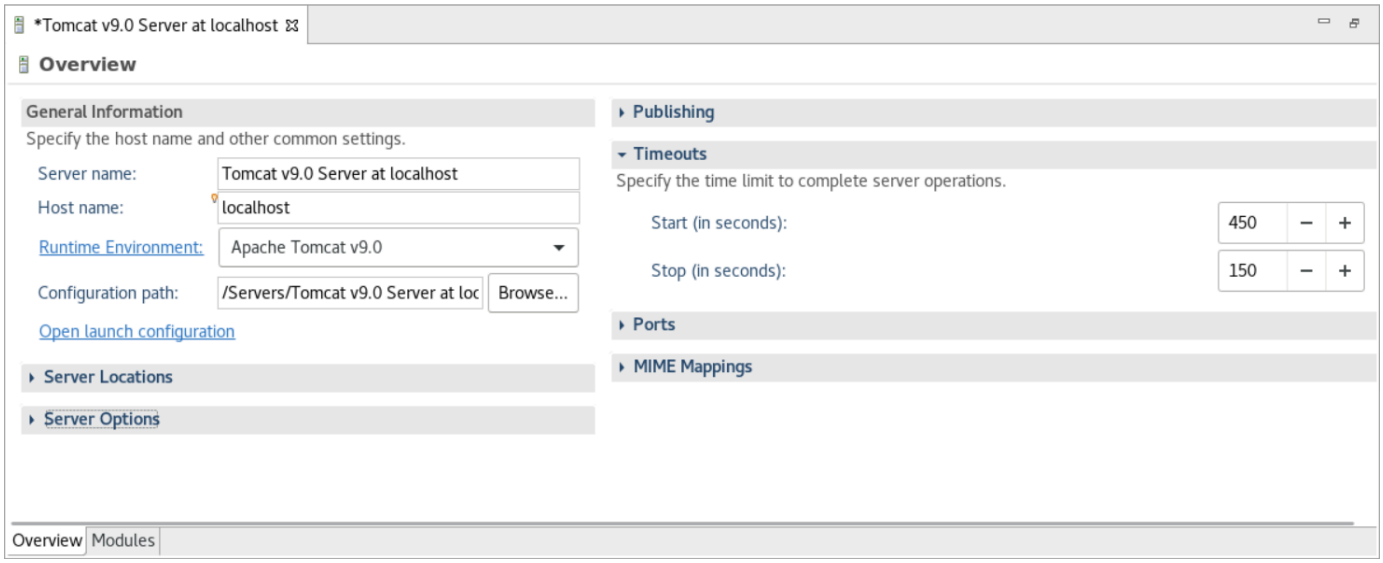
1. 選擇該`PlanetsDemo-web`文件，然後選擇運行方式 > Maven 安裝。`PlanetsDemo-web`再次選取並選擇「重新整理」，以確保 nPM 編譯的前端已正確編譯為 .war 並被 Eclipse 注意到。
2. 將 PlanetsDemo-runtime.zip 上傳至執行個體，然後在可存取的位置解壓縮檔案。這可確保示範應用程式可以存取所需的設定資料夾和檔案。
3. `PlanetsDemo-runtime/tomcat-config`將的內容複製到您為 Tomcat 伺服器建立的`Servers/Tomcat v9.0...`子資料夾中：



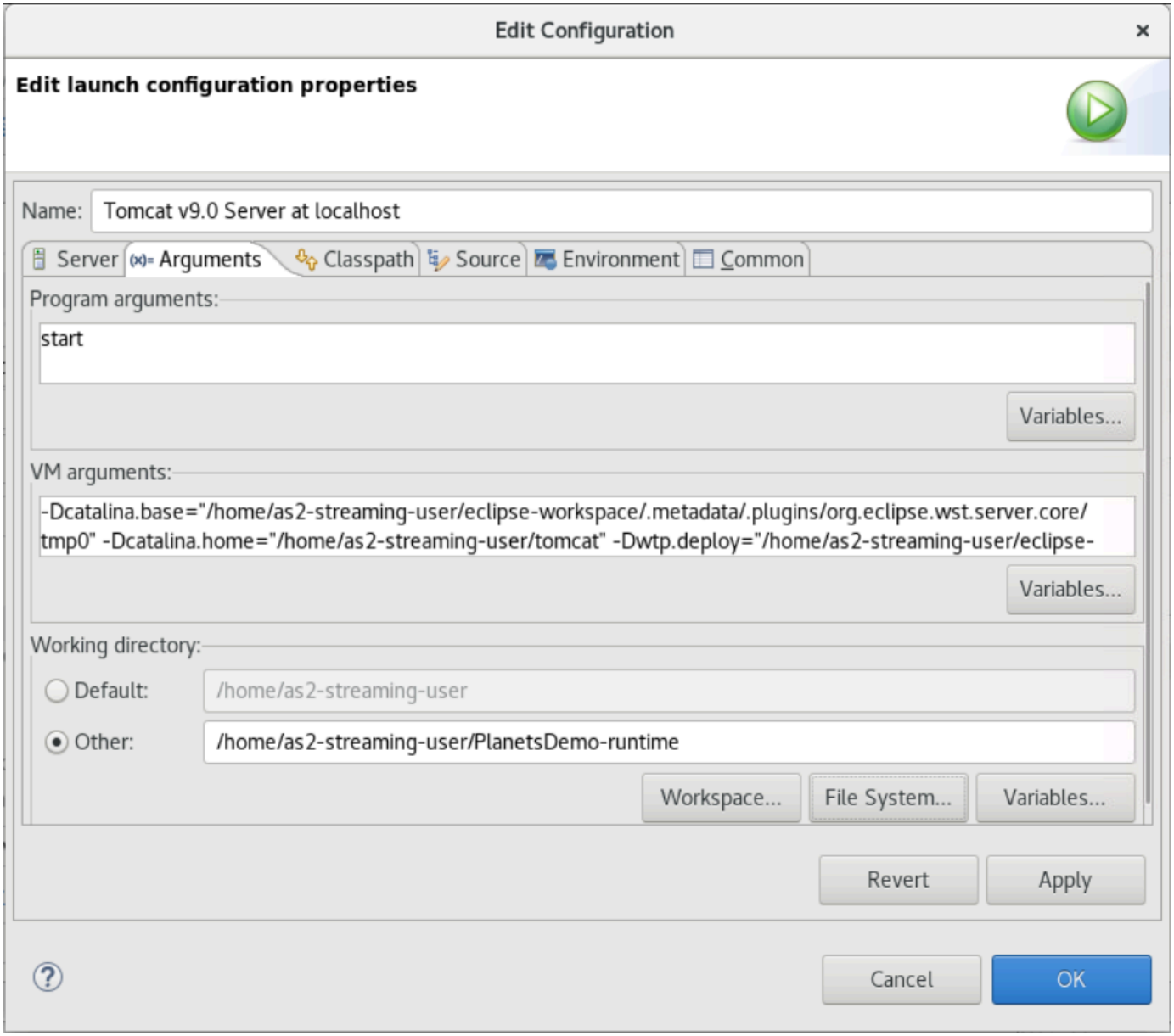
4. 在「伺`tomcat v9.0`服器」檢視中開啟伺服器項目。伺服器屬性編輯器隨即出現：



5. 在 [概觀] 索引標籤中，將 [開始] 的 [逾時] 值增加到 450 秒，對於 [停止]，將 [逾時] 值增加為 150 秒，如下所示：



6. 選擇 [開啟啟動設定]。這時系統顯示嚮導。在精靈中，導覽至「參數」資料夾，並針對「工作目錄」選擇「其他」。選擇文件系統，然後導航到先前解壓縮的文件`PlanetsDemo-runtime`夾。此資料夾應包含名為 config 的直接子資料夾。

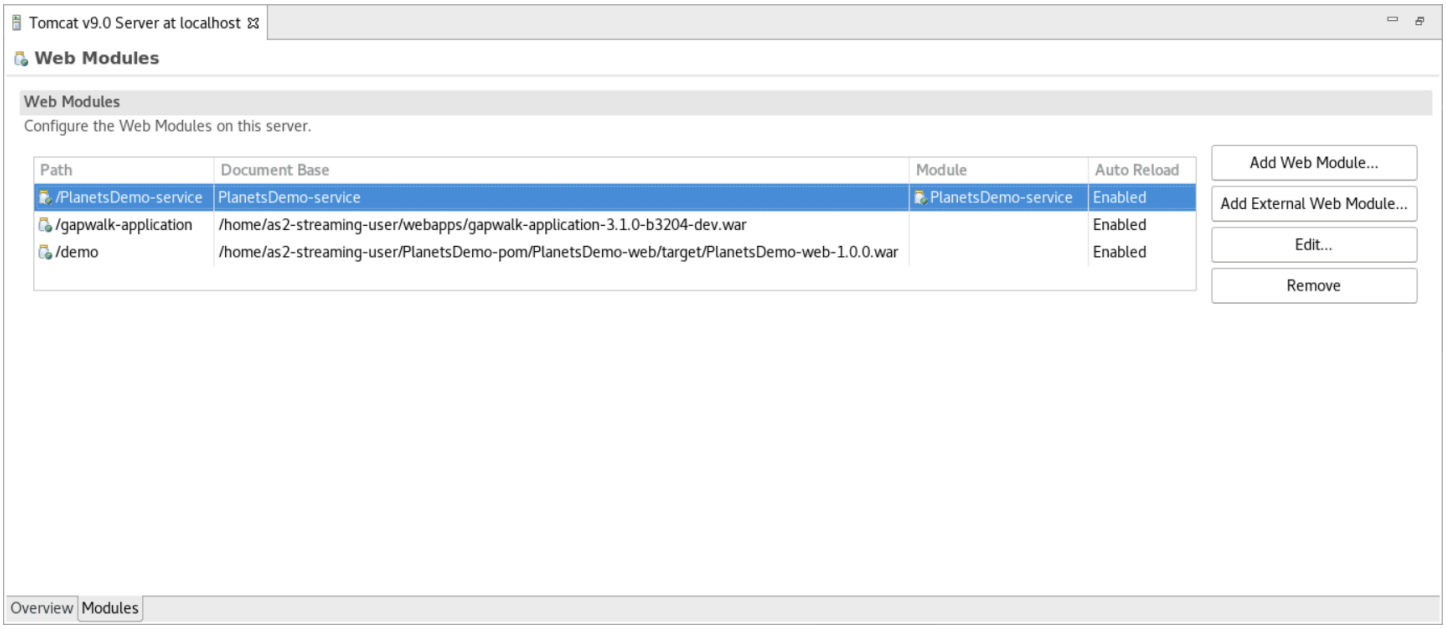


7. 選擇伺服器屬性編輯器的「模組」索引標籤，然後進行下列變更：

   - 選擇添加 Web 模塊並添加`PlanetsDemo-service`。
   - 選擇「新增外部 Web 模組」。這時系統顯示「添加 Web 模塊」對話框。進行下列變更：
     - 在「文件庫」中，選擇「瀏覽」並導覽至 `~/webapps/gapwalk-application...war`
     - 在路徑中，輸入`/gapwalk-application`。
   - 選擇確定。
   - 再次選擇「新增外部 Web 模組」，然後進行下列變更：
     - 在「文件庫」中，輸入前端 .war (在中) 的路徑 `PlanetsDemo-web/target`

- 對於「路徑」，輸入 /demo
- 選擇 OK (確定)。
- 儲存編輯器所做的修改 (Ctrl + S)。

編輯器內容現在應該類似於以下內容。



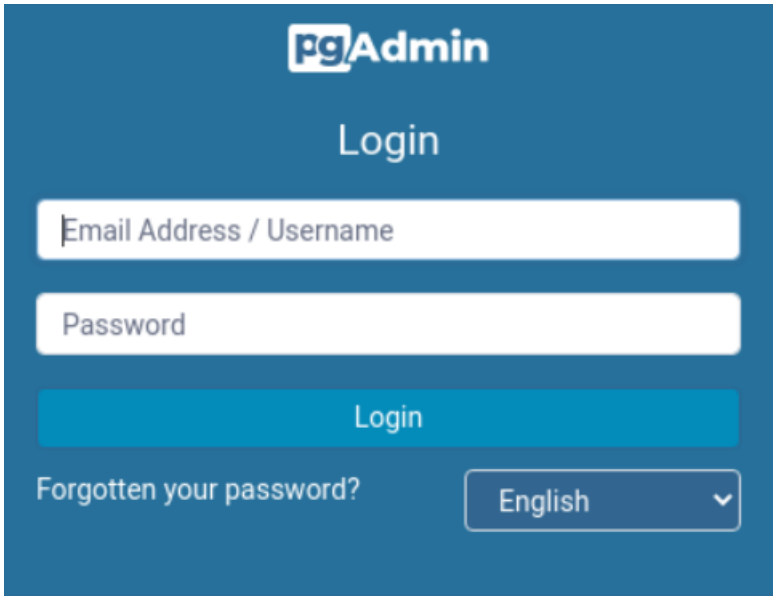
步驟 8：建立資JICS料庫

在此步驟中，您會連線至您在中建立的資料庫步驟 1：建立資料庫。

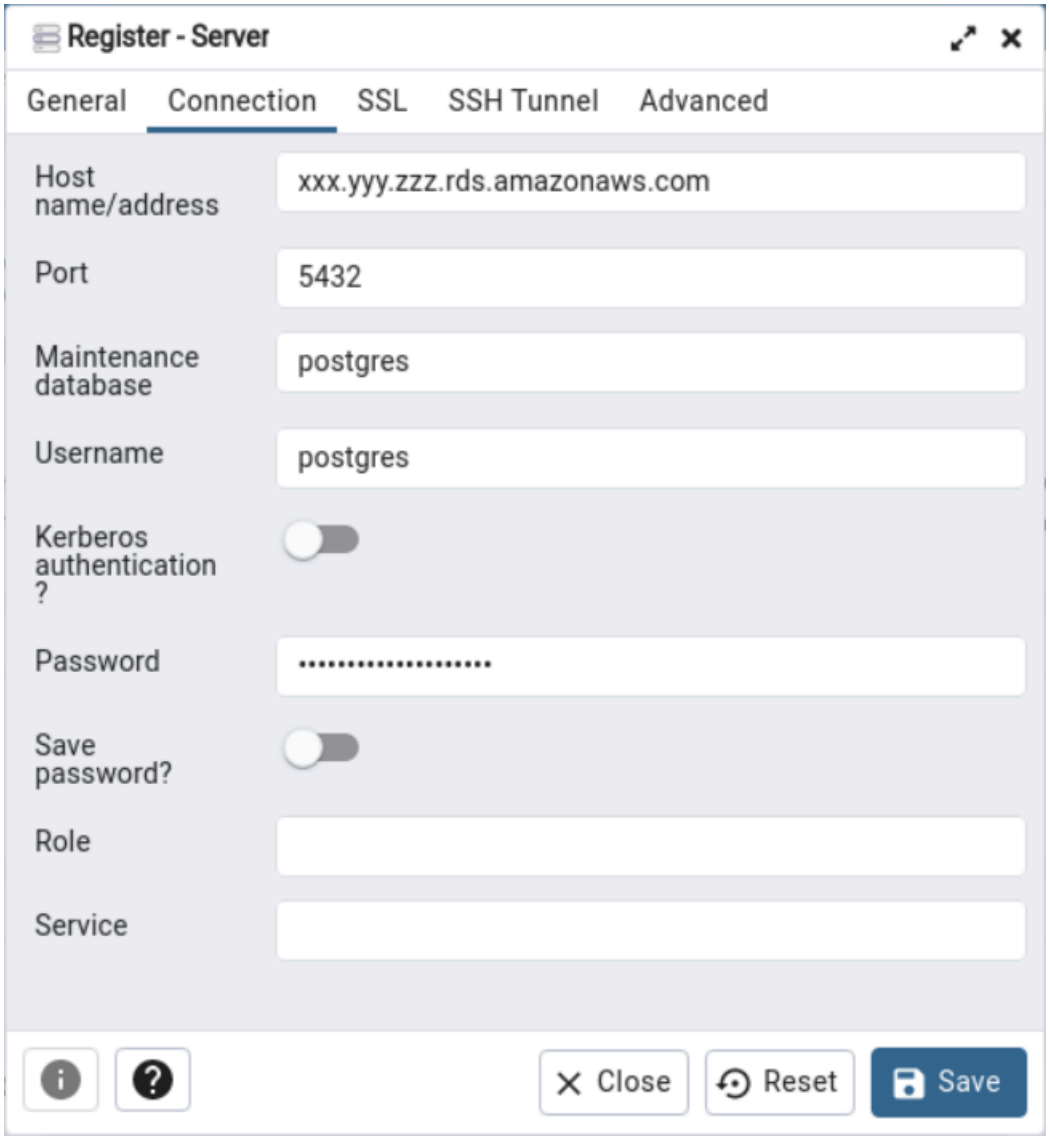
1. 從 AppStream 2.0 實例中，在終端機中發出以下命令以啟動pgAdmin：

```
./pgadmin-start.sh
```

2. 選擇一個電子郵件地址和密碼作為登錄標識符。注意提供的內容URL（通常為 http://127.0.0.1:5050）。在實例中啟動 Google Chrome 瀏覽器，將其複製並粘貼URL到瀏覽器中，然後使用您的標識符登錄。



3. 登入之後，請選擇「新增伺服器」，然後輸入先前建立之資料庫的連線資訊，如下所示。

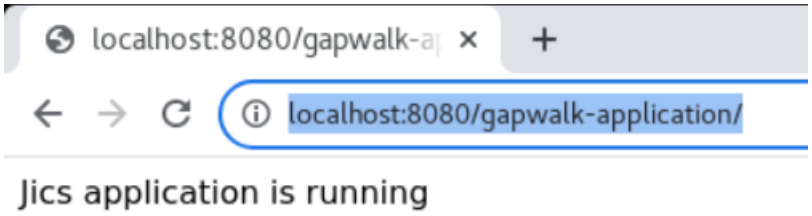


4. 當您連線到資料庫伺服器時，請使用「物件」>「建立」>「資料庫」，並建立名為 jics 的新資料庫。
5. 編輯示範應用程式使用的資料庫連線資訊。此資訊在中定義`PlanetsDemo-runtime/config/application-main.yml`。搜尋項`jicsDs`目。若要擷取`username`和的值`password`，請在 Amazon 主RDS控台中導覽至資料庫。在組態索引標籤的主要認證下ARN，選擇在 Secrets Manager 中管理。然後，在密碼管理員主控台中，選擇 [擷取密碼值]。

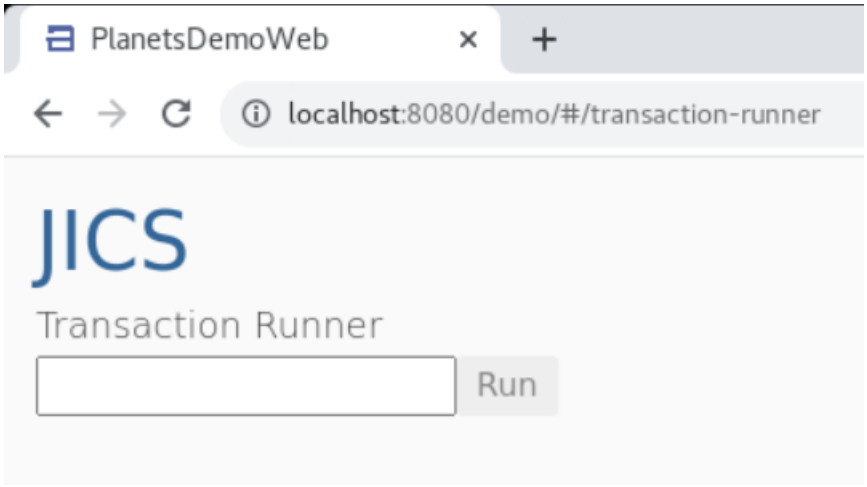
步驟 9：啟動並測試應用程式

在此步驟中，您將啟動 Tomcat 伺服器和示範應用程式，以便您可以對其進行測試。

1. 若要啟動 Tomcat 伺服器和先前部署的應用程式，請在 [伺服器] 檢視中選取伺服器項目，然後選擇 [啟動]。隨即出現顯示啟動記錄的主控台。
2. 在「伺服器」檢視中檢查伺服器狀態，或等待主控台中的 [xxx] 毫秒內「伺服器啟動」訊息。服務器啟動後，檢查間隙行程應用程序是否正確部署。要做到這一點，訪問 http://localhost:8080/gapwalk-application URL 在谷歌瀏覽器瀏覽器。您應該會看到以下內容。



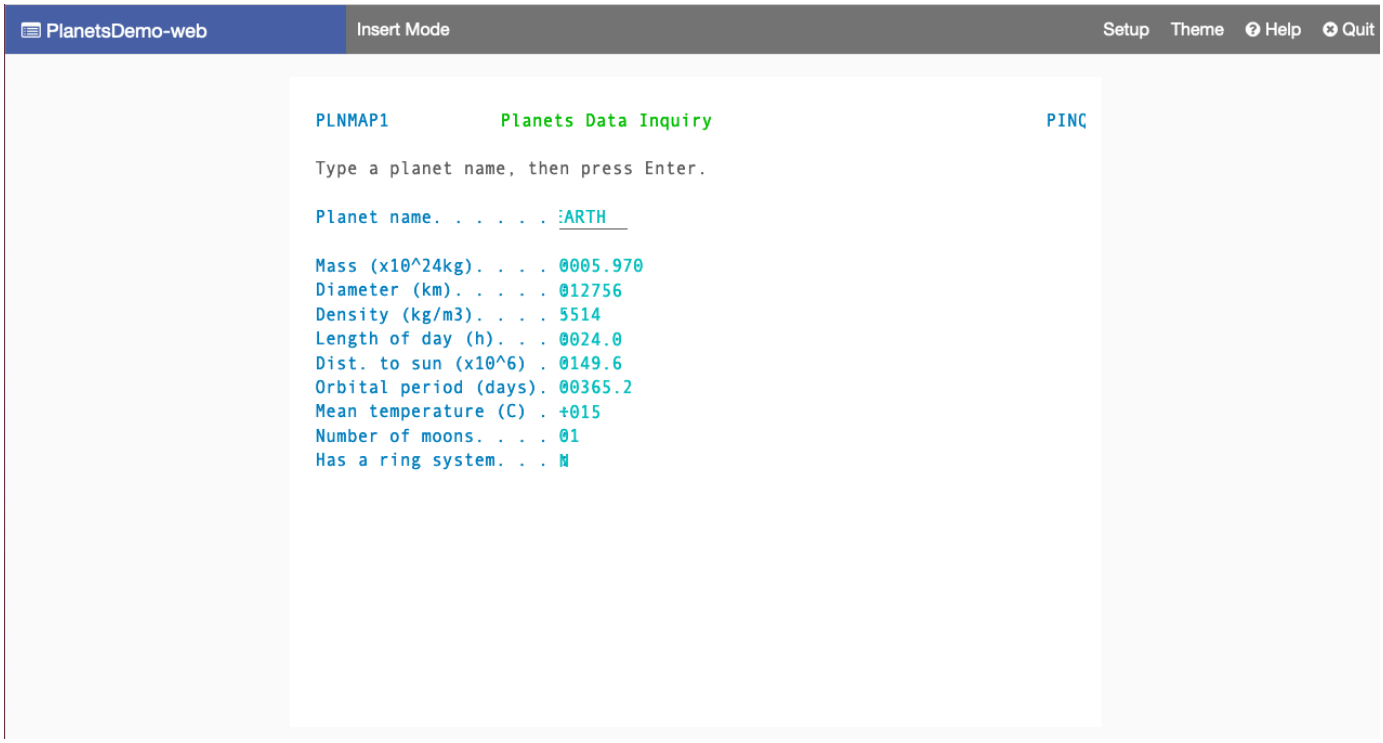
3. 從谷歌瀏覽器訪問已部署的應用程序前端 http://localhost:8080/demo. 下列 [交易啟動器] 頁面應該會出現。



4. 若要啟動應用程式交易，請PINQ在輸入欄位中輸入，然後選擇「執行」(或按 Enter)。

   示範應該會出現應用程式畫面。

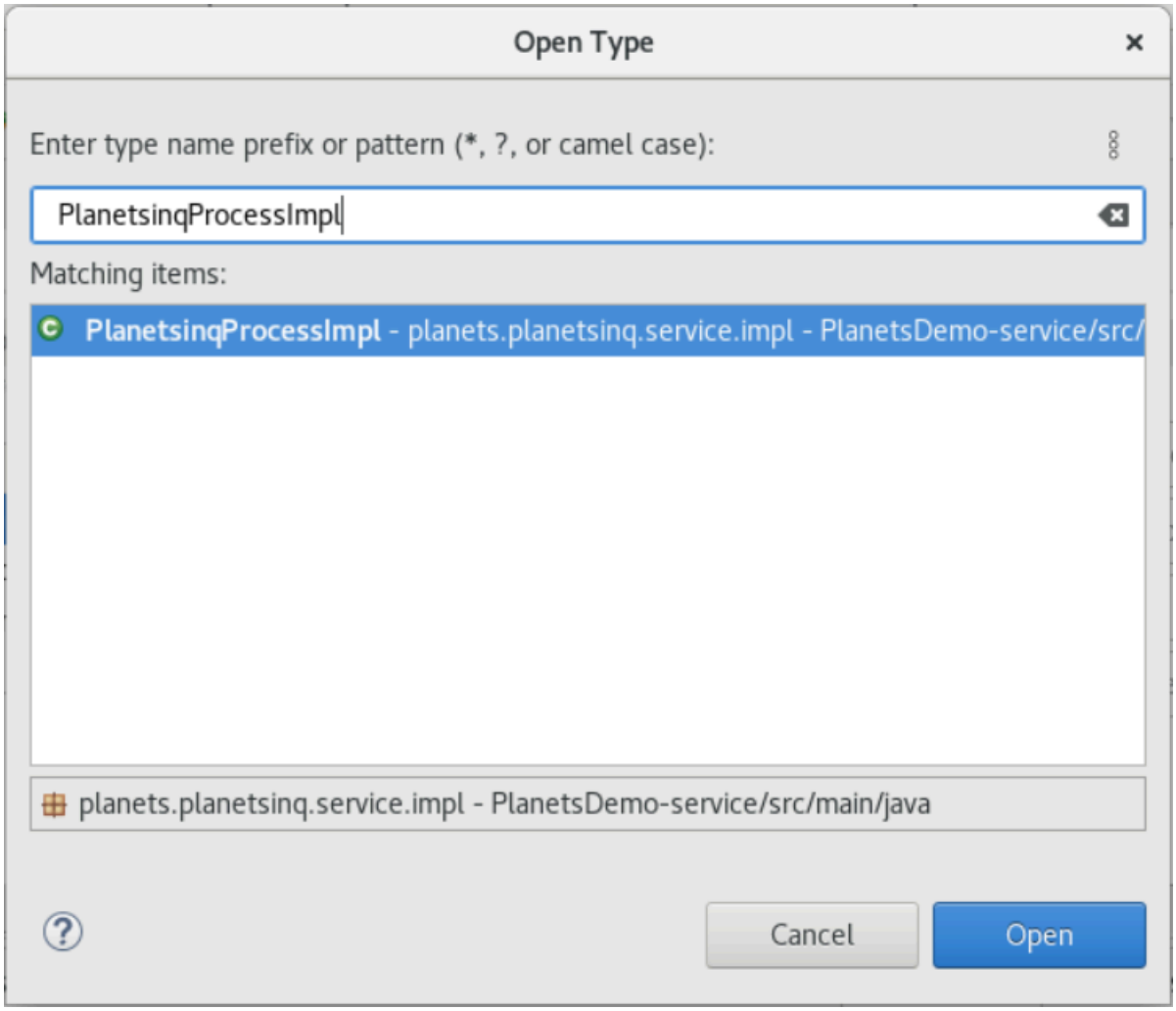
5. 在對應的欄位中輸入行星名稱，然後按 Enter。



步驟 10：調試應用程序

在此步驟中，您可以使用標準的 Eclipse 偵錯功能進行測試。當您使用現代化的應用程式時，就可以使用這些功能。

1. 若要開啟主要服務類別，請按住 Ctrl + 轉移 + T。然後輸入`PlanetsinqProcessImpl`。



2. 導航到該`searchPlanet`方法，並在那裡放置一個斷點。
3. 選取伺服器名稱，然後選取在偵錯中重新啟動。
4. 重複前面的步驟。也就是說，訪問應用程序，輸入行星名稱，然後按 Enter 鍵。

   Eclipse 會在該`searchPlanet`方法中停止應用程序。現在您可以檢查它。

清除資源

如果您不再需要為此教學課程建立的資源，請刪除這些資源，以免產生額外費用。請完成下列步驟：

- 如果行星應用程序仍在運行，請停止它。
- 刪除您在中建立的資料庫步驟 1：建立資料庫。如需詳細資訊，請參閱刪除資料庫執行個體。

使用微焦點重新平台應用

本指南涵蓋使用上的大型主機現代化解決方案重新平台大型主機應用 end-to-end AWS 程式的程序。AWS它描述 AWS 了所有任務，並包括在 Amazon 上設定和操作大型主機現代化執行時期的相關資訊，EC2從初始設定和分析到在其上建置、測試和部署現代化應用程式。 AWS它還涵蓋了進階主題，例如使用舊式資料結構、使用範本和預先定義的專案，以及為串流工作階段設定自動化。

主題

- 設置微型聚焦運行時間（在 Amazon 上EC2）
- 為 Micro Focus 企業分析儀和 Micro Focus 企業開發人員串流會議設定自動化
- 在企業開發人員中以表格和欄的形式檢視資料集
- 微焦點教程
- AWS 大型主機現代化中可用的批次公用程式

設置微型聚焦運行時間（在 Amazon 上EC2）

AWS Mainframe Modernization 提供數個 Amazon 機器映像 (AMIs)，其中包括微焦點授權的產品。這些功能可AMIs讓您快速佈建 Amazon 彈性運算雲端 (AmazonEC2) 執行個體，以支援您控制和管理的 Micro Focus 環境。本主題提供存取和啟動這些步驟所需的步驟AMIs。使用這些AMIs是完全可選的，並且不需要完成本使用指南中的自學課程。

主題

- 設置微型聚焦運行時的先決條件（在 Amazon 上EC2）
- 為 Amazon S3 創建亞馬遜VPC端點
- 要求帳戶的允許清單更新
- 建立角 AWS Identity and Access Management 色
- 授予 License Manager 所需的權限
- 訂閱 Amazon 機器映像
- 啟動 AWS Mainframe Modernization 微焦點實體
- 子網路或沒VPC有網際網路存取

設置微型聚焦運行時的先決條件（在 Amazon 上EC2）

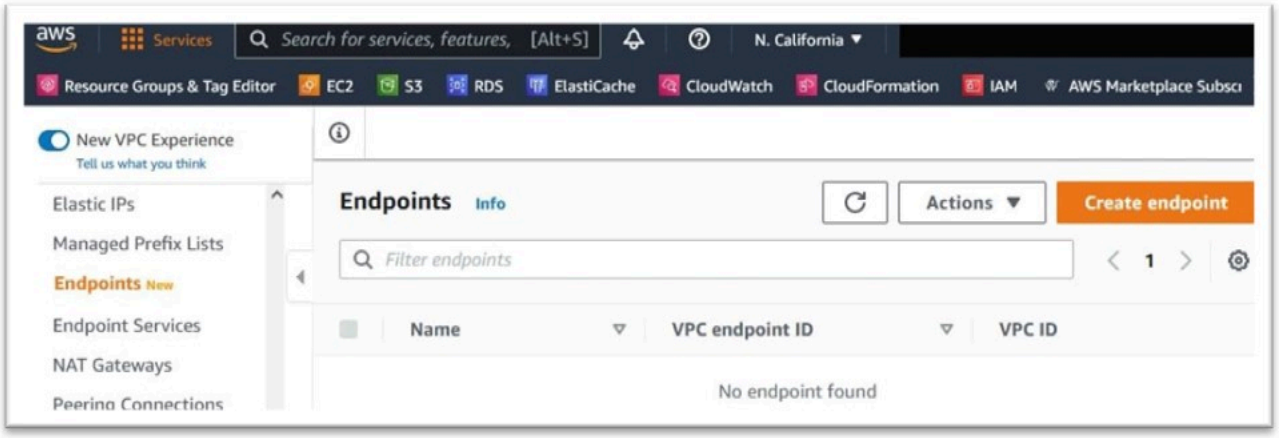
當您設定 Micro Focus 執行階段 (在 Amazon 上EC2) 時，請確定您符合下列先決條件。

- 管理員可存取將在其中建立 Amazon EC2 執行個體的帳戶。
- 識別 AWS 區域 要建立 Amazon EC2 執行個體的位置，並確認 AWS Mainframe Modernization 服務是否可用。請參閱AWS 服務 (按地區)。請務必選擇可使用該服務的地區。
- 識別將在其中建立 Amazon EC2 執行個體的 Amazon Virtual Private Cloud (AmazonVPC)。

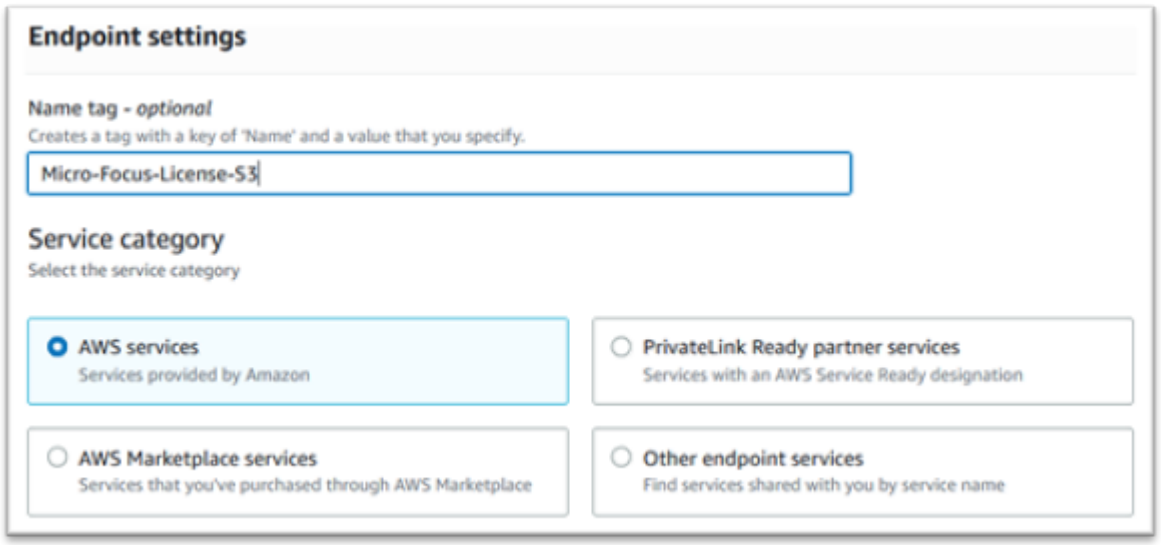
為 Amazon S3 創建亞馬遜VPC端點

在本節中，您將建立供 Amazon S3 使用的亞馬遜VPC端點。設定此端點將在稍後設定網際網路存取時協助您VPC。

1. 導航到 Amazon VPC 在 AWS Management Console.
2. 在導覽窗格中選擇端點。
3. 選擇建立端點。



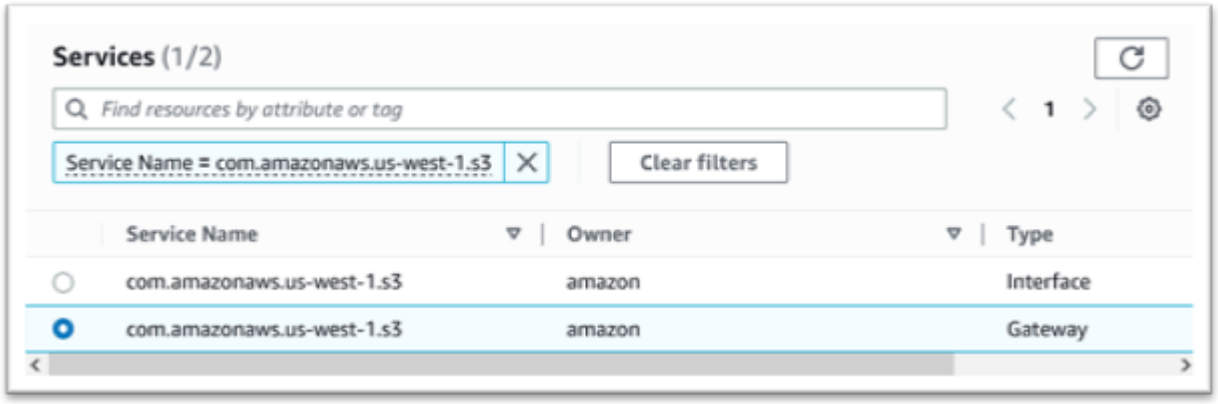
4. 輸入有意義的名稱標籤，例如：「微焦點-授權-S3」。
5. 選擇「AWS服務」作為「服務類別」。



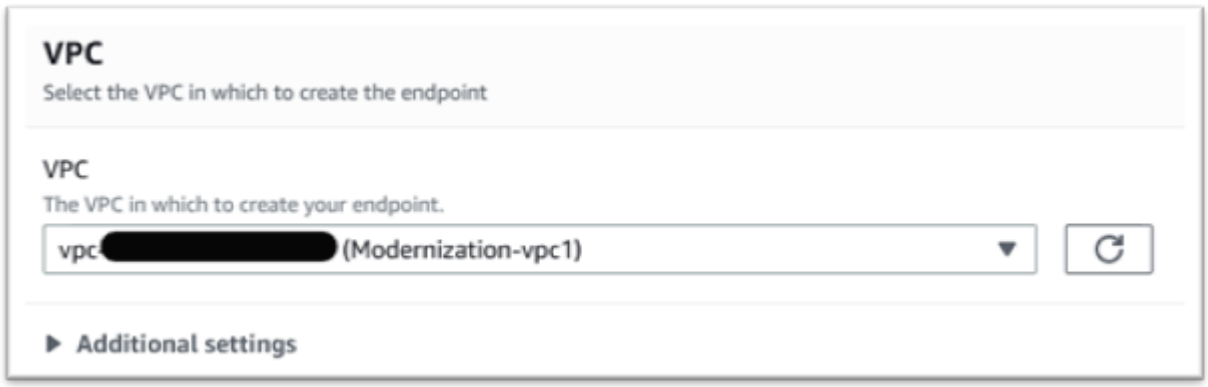
6. 在「服務」下搜尋 Amazon S3 閘道服務： [區域] .s3。

   對於us-west-1這將是：com.amazonaws.us-west-1.s3。

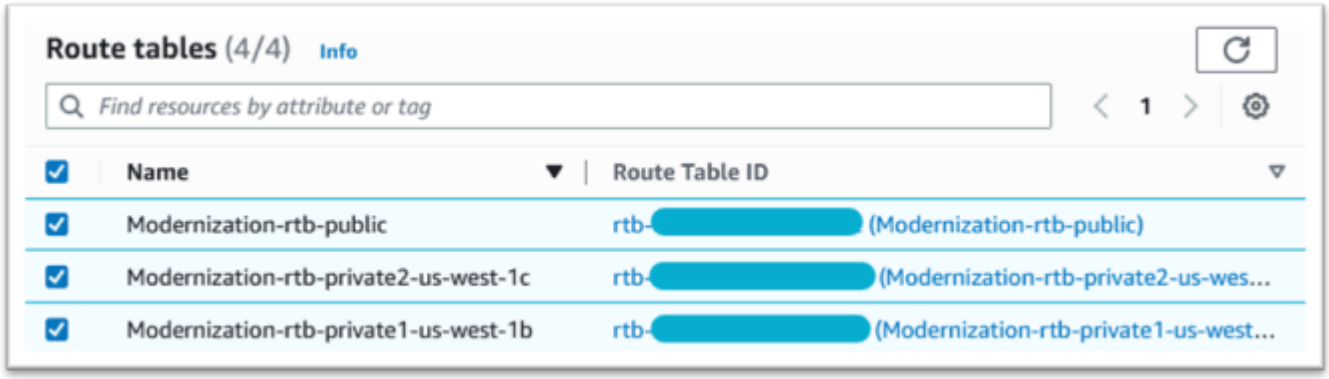
7. 選擇閘道服務。



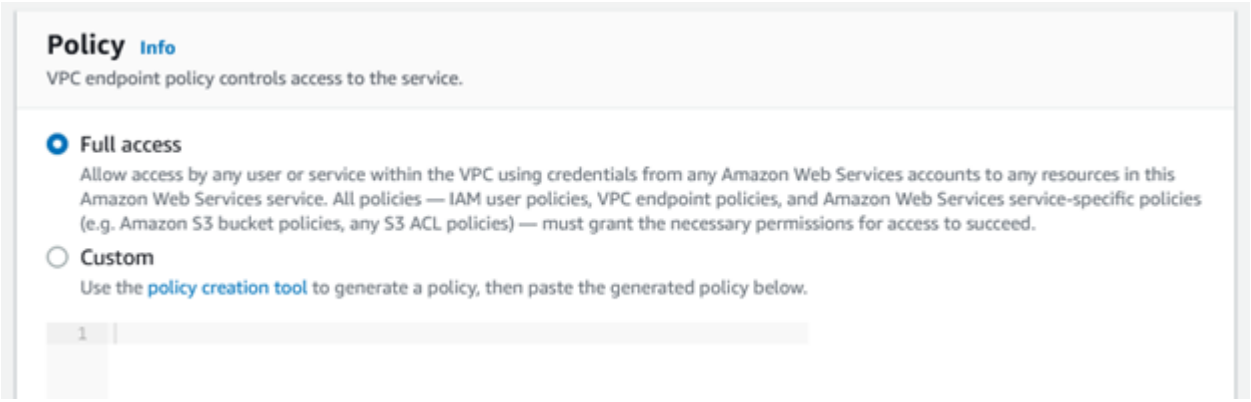
8. 對於VPC選擇VPC您將要使用的。



9. 選擇的所有路由表VPC。



10. 在「策略」下選擇「完整存取



11. 選擇建立端點。

要求帳戶的允許清單更新

請與您的 AWS 代表合作，讓您的帳戶允許列出 AWS Mainframe Modernization AMIs。請提供以下資料：

- AWS 帳戶 識別碼。
- Amazon VPC 端 AWS 區域 點的創建位置。
- 在中建立的 VPC Amazon S3 端點識別碼為 Amazon S3 創建亞馬遜VPC端點。這是公司的vpce-xxxxxxxxxxxxxxxxx代碼。 [區域] .s3 閘道端點。
- 所有 Micro Focus 企業套件 AMI Amazon EC2 執行個體所需的授權數量。

  每個CPU核心需要一個授權 (大多數 Amazon EC2 執行個體 vCPUs 為每 2 個授權)。

  如需詳細資訊，請參閱最佳化CPU選項。

  請求的數量可以在將 future 通過調整 AWS。

> **Note**
>
> AWS 代表必須開啟允許清單要求的支援票證。無法直接請求，並且請求可能需要幾天的時間才能完成。

建立角 AWS Identity and Access Management 色

建立供 AWS Mainframe Modernization Amazon EC2 執行個體使用的 AWS Identity and Access Management 政策和角色。透過IAM主控台建立角色會建立相同名稱的關聯執行個體設定檔。將此執行個體設定檔指派給 Amazon EC2 執行個體可讓您指派 Micro Focus 授權。如需執行個體設定檔的詳細資訊，請參閱使用IAM角色將許可授與在 Amazon EC2 執行個體上執行的應用程式。

建立IAM策略

會先建立IAM原則，然後再附加至角色。

1. 導覽至 AWS Identity and Access Management 中的 AWS Management Console。
2. 選擇策略，然後選擇創建策略。

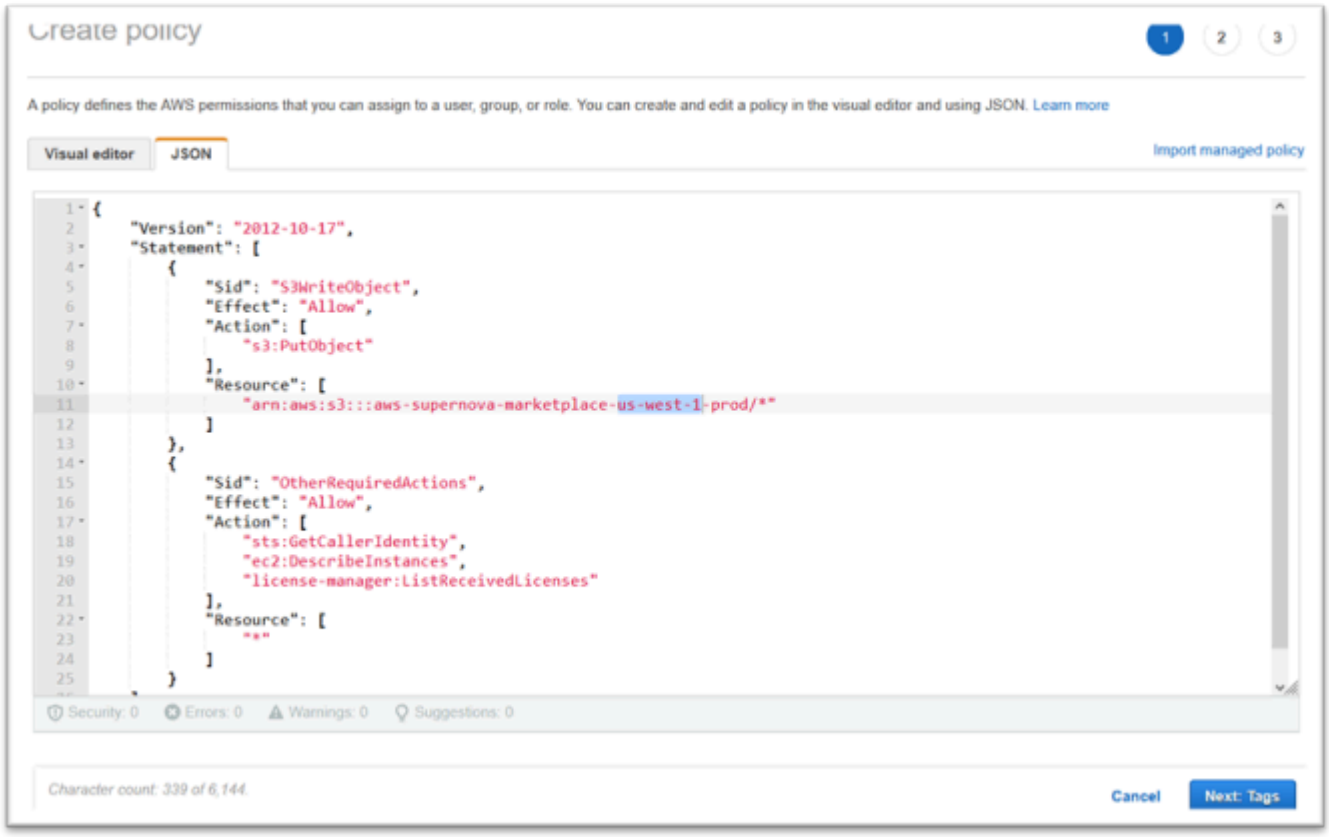
3. 選擇索JSON引標籤。

4. 將以下us-west-1JSON內容取代為定義 Amazon S3 端點的 AWS 區域 位置，然後將其複製並貼JSON到政策編輯器中。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3WriteObject",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::aws-supernova-marketplace-us-west-1-prod/*"
            ]
        },
        {
            "Sid": "OtherRequiredActions",
            "Effect": "Allow",
            "Action": [
                "sts:GetCallerIdentity",
                "ec2:DescribeInstances",
                "license-manager:ListReceivedLicenses"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
```

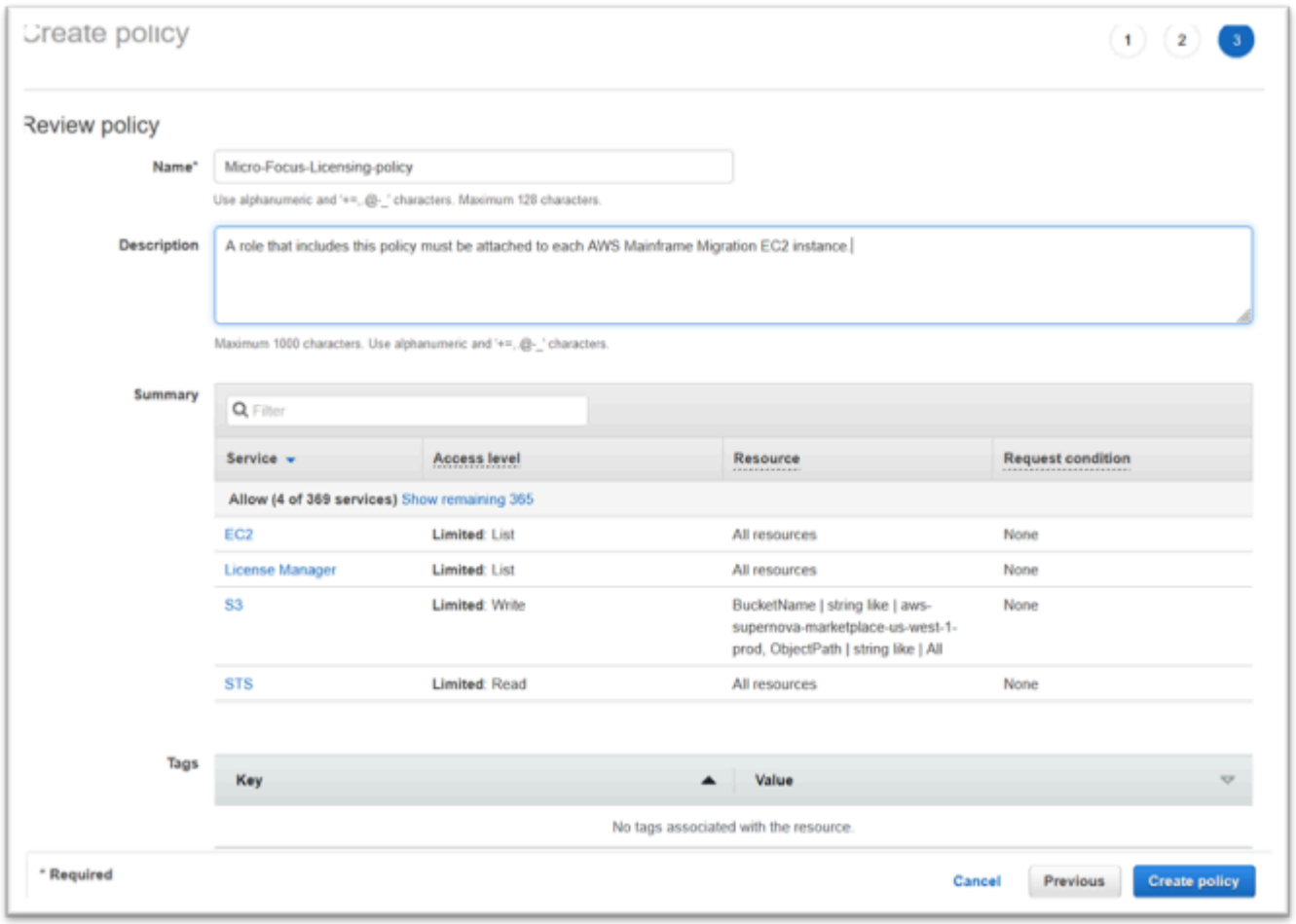
```
}
```

> **Note**
>
> Sid 下的動作`OtherRequiredActions`不支援資源層級權限，必須在資源元素`*`中指定。



5. 選擇下一步：標籤。

6. 選擇性地輸入任何盤點單，然後選擇下一步：複查。
7. 輸入策略的名稱，例如「微焦點-授權政策」。選擇性地輸入說明，例如「必須將包含此政策的角色附加到每個 AWS Mainframe Modernization Amazon EC2 執行個體。」

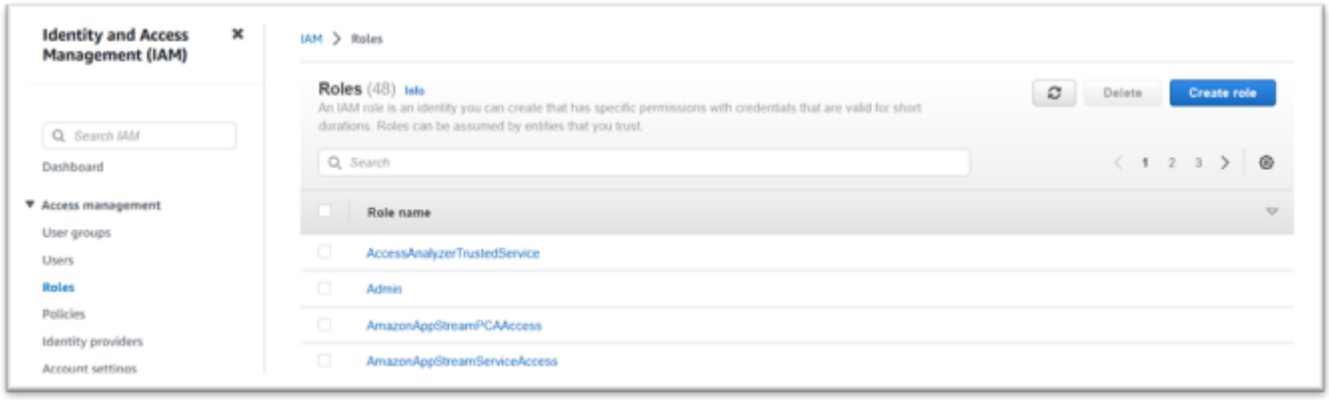


8. 選擇 建立政策 。

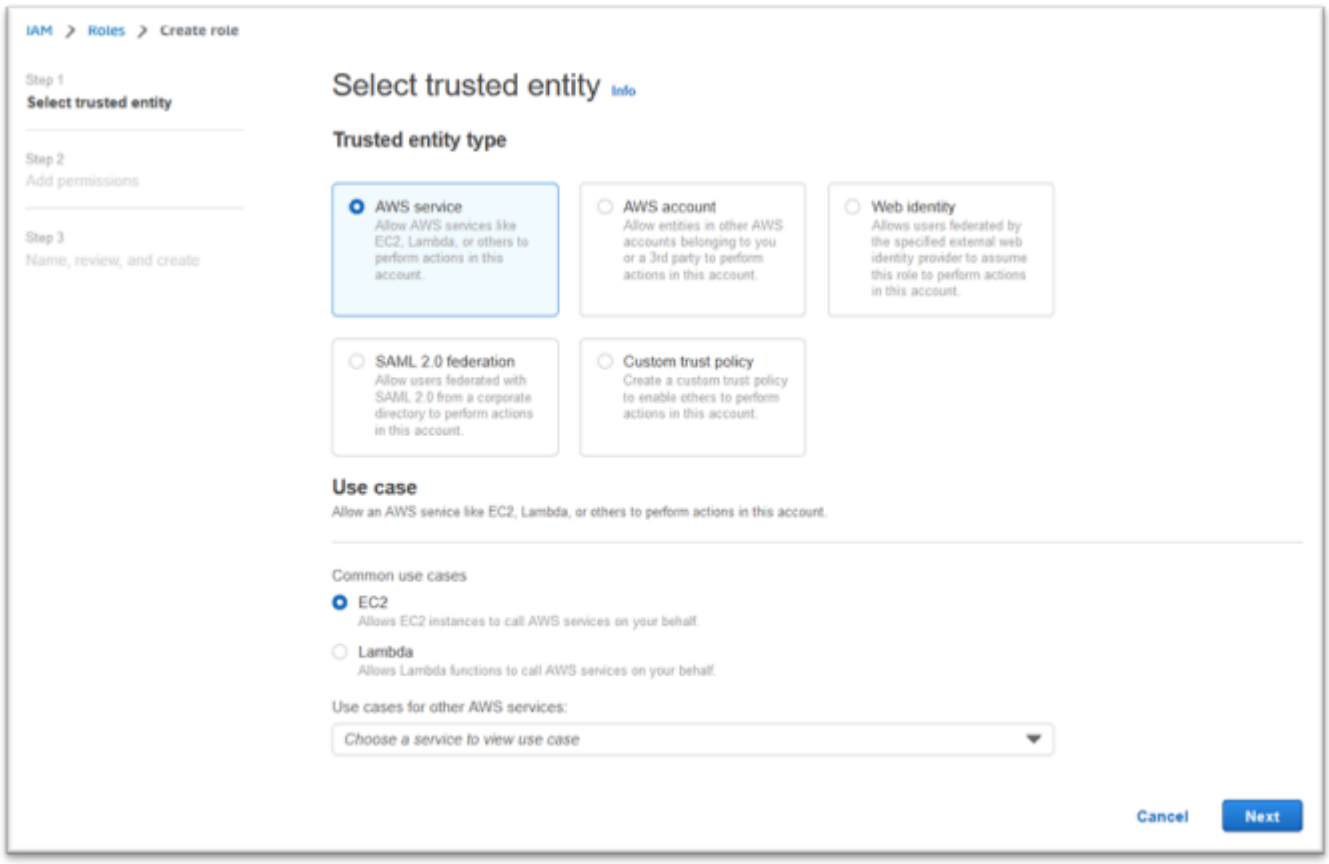
建立角IAM色

建立IAM策略之後，您可以建立IAM角色並將其附加至策略。

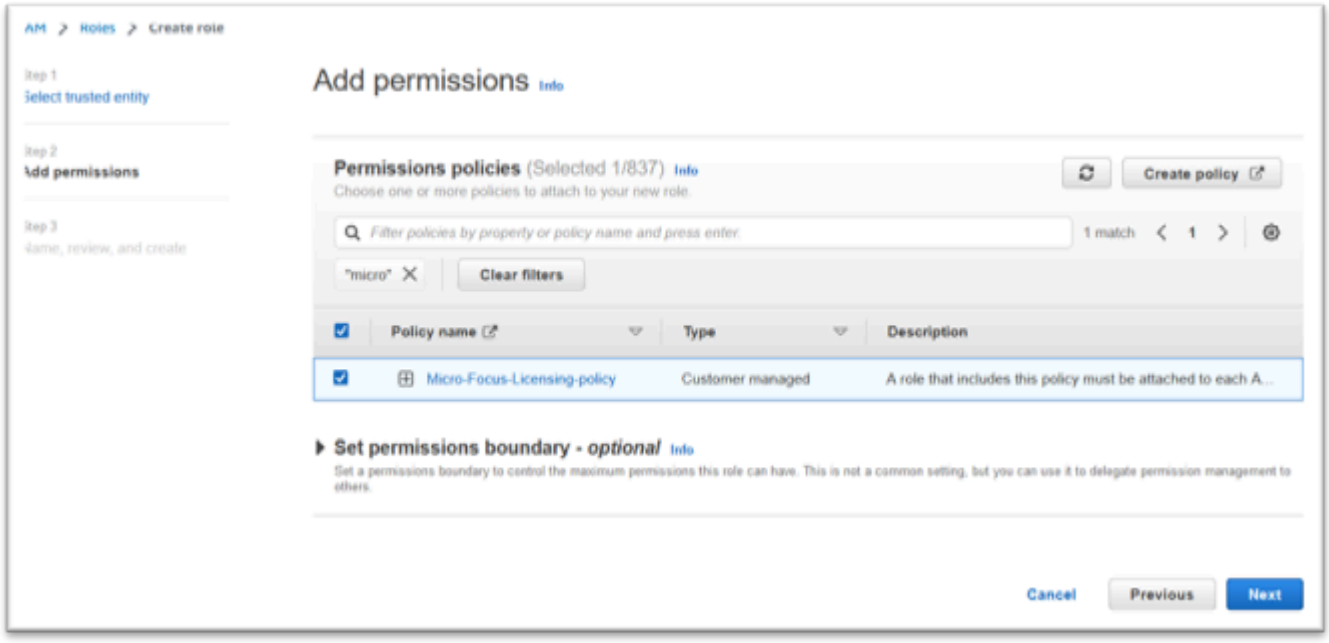
1. 導覽至IAM中的 AWS Management Console。
2. 選擇角色，然後選擇建立角色。



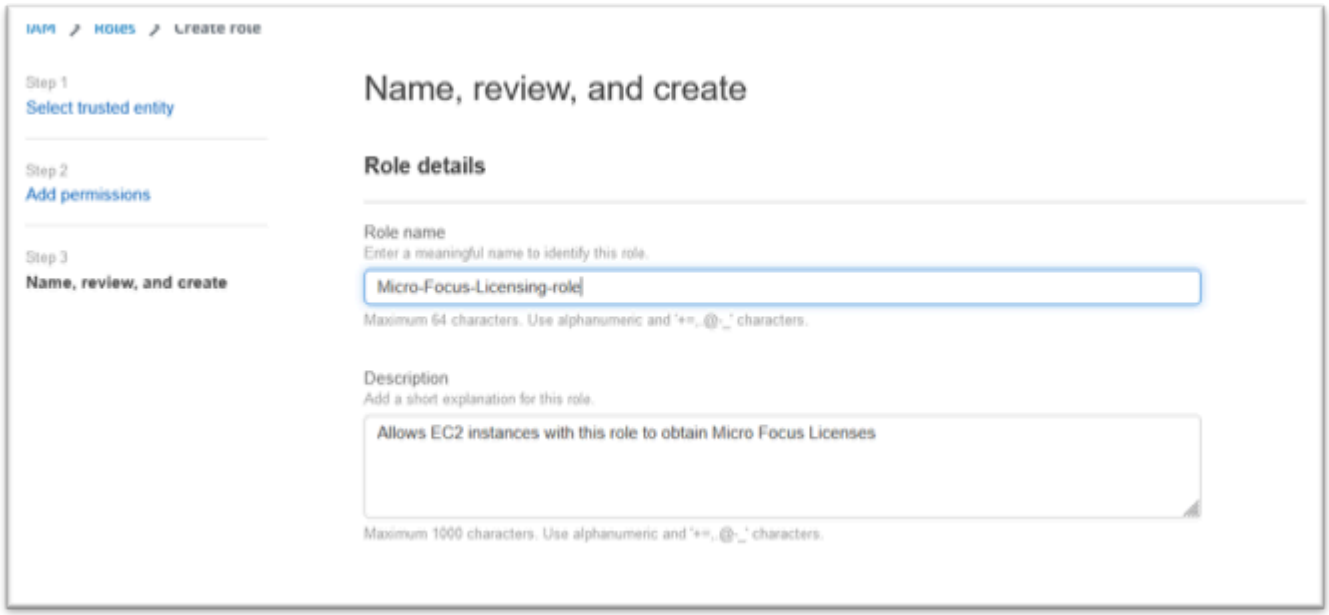
3. 將信任的實體類型保留為AWS 服務，然後選擇EC2常見使用案例。



4. 選擇 Next (下一步)。
5. 在過濾器中輸入「微型」，然後按 Enter 鍵以應用過濾器。
6. 選擇剛剛建立的政策，例如「微焦點授權政策」。
7. 選擇 Next (下一步)。



8. 輸入「角色」名稱，例如「微焦點-授權-角色」。
9. 將說明取代為您自己的說明，例如「允許具有此角色的 Amazon EC2 執行個體取得 Micro Focus 授權」。



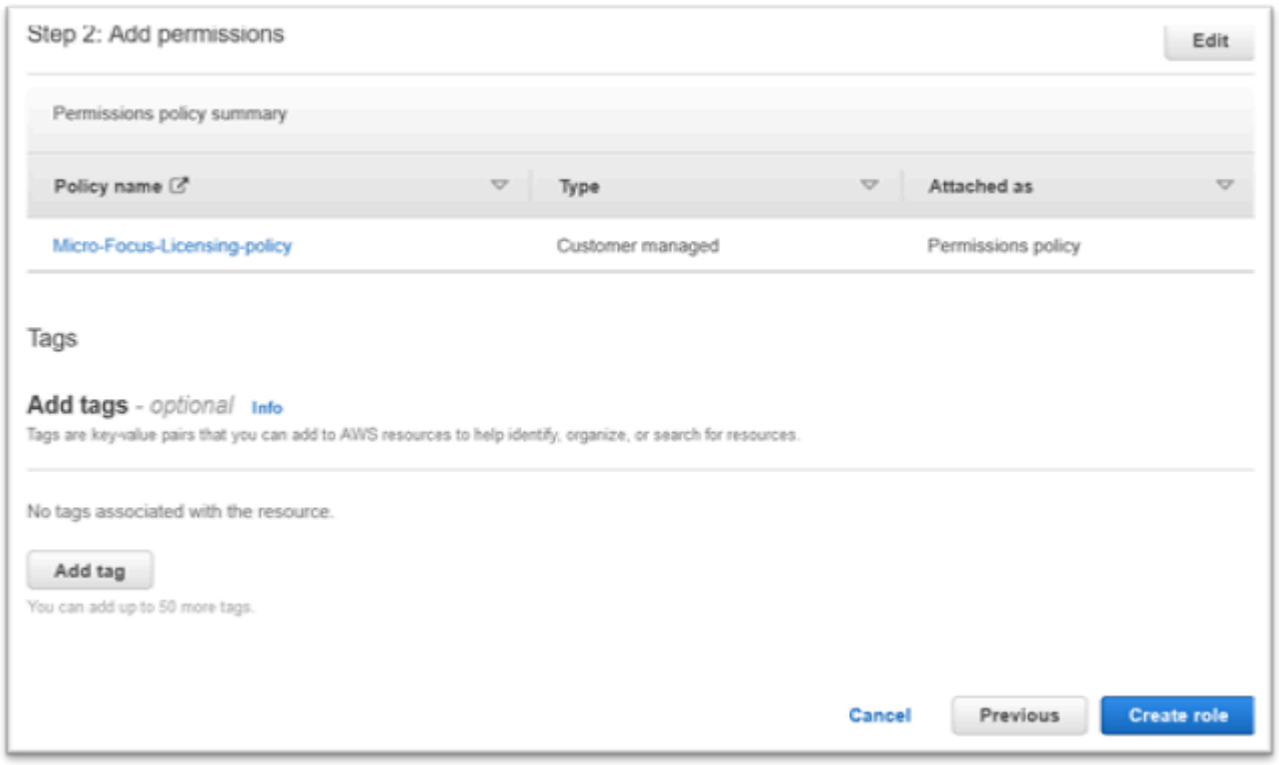
10. 在 [步驟 1：選取受信任的實體] 下，檢閱JSON並確認其具有下列值：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Principal": {
                "Service": [
                    "ec2.amazonaws.com"
                ]
            }
        }
    ]
}
```

> **Note**
>
> 效果，行動和主要的順序並不重要。

11. 確認「步驟 2：新增權限」 會顯示您的授權原則。



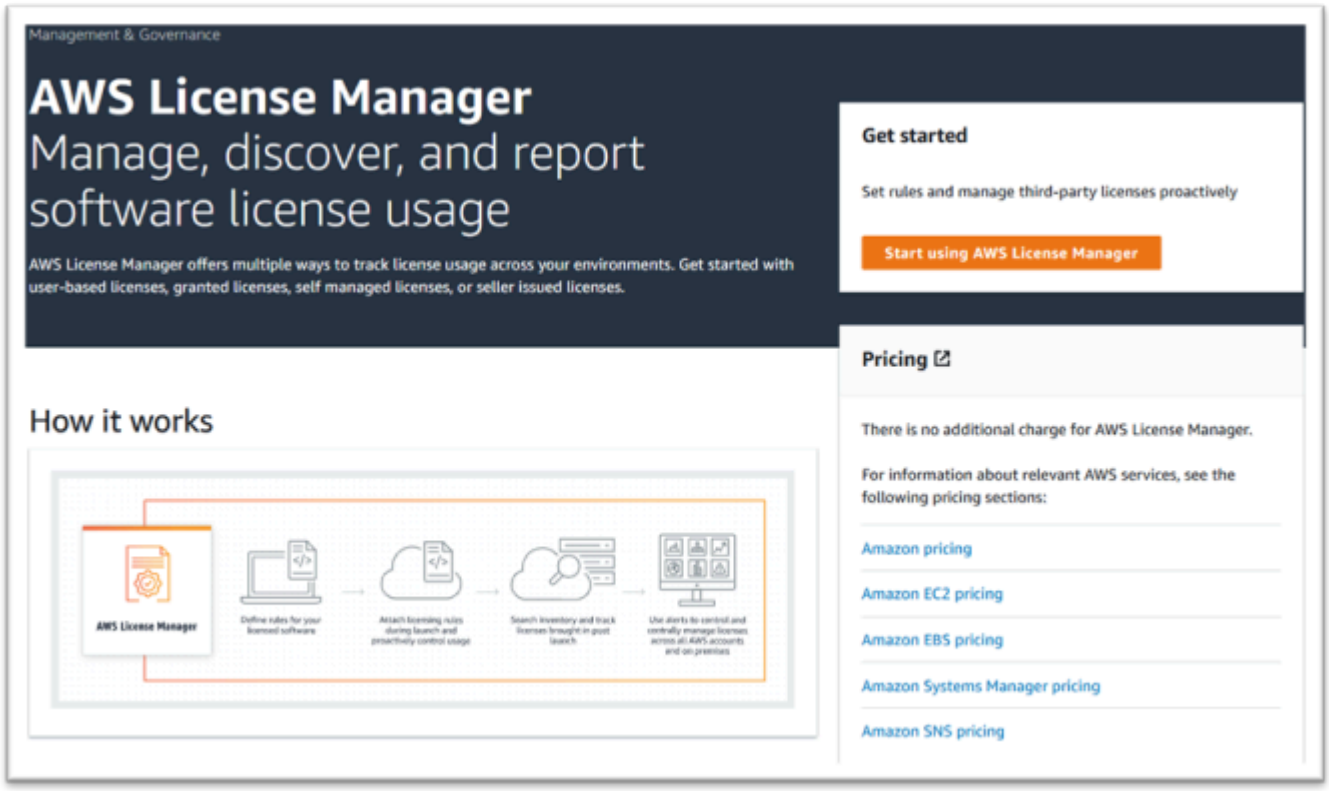
12. 選擇建立角色。

允許清單要求完成後，請繼續執行下列步驟。

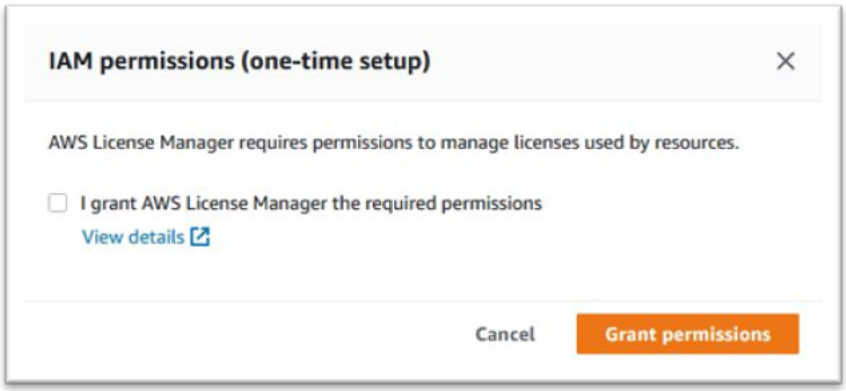
授予 License Manager 所需的權限

您需要授予許可， AWS License Manager 以設置 Micro Focus 運行時引擎（在 Amazon 上EC2）。

1. 導覽至 AWS License Manager 中的 AWS Management Console。



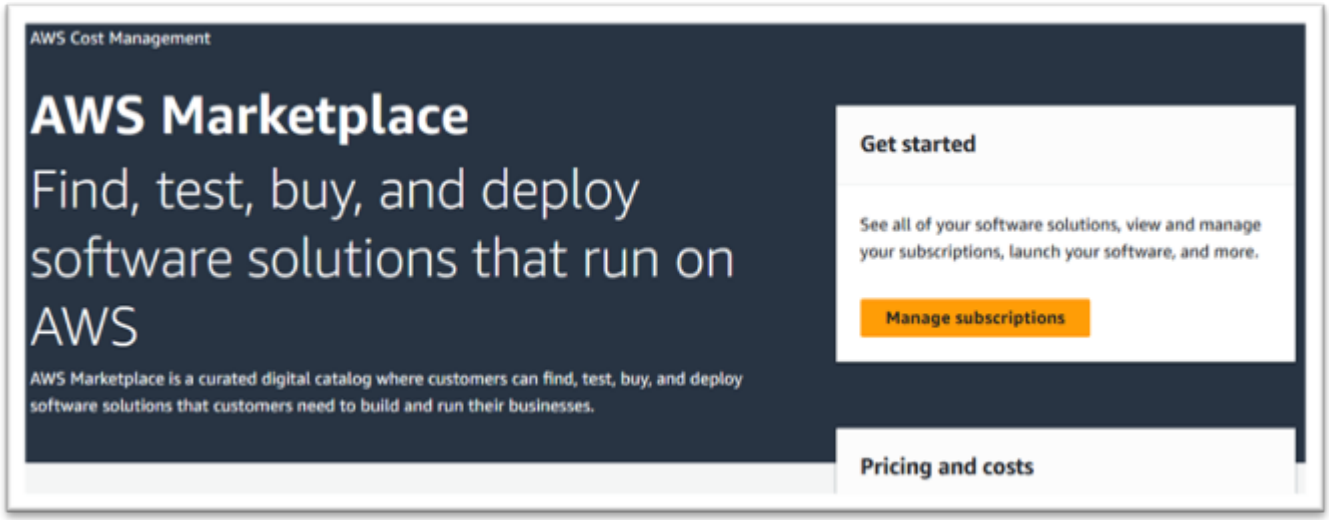
2. 選擇開始使用 AWS License Manager。
3. 如果您看到以下彈出窗口，請查看詳細信息，然後選擇複選框並按授予權限。



訂閱 Amazon 機器映像

訂閱 AWS Marketplace 產品後，您可以從產品啟動執行個體AMI。您也可以在設定 Micro Focus 執行階段引擎AMIs時管理您的訂閱項目 (在 Amazon 上EC2)。

1. 導覽至中的「 AWS Marketplace 訂閱」 AWS Management Console。
2. 選擇 Manage subscriptions (管理訂閱)。



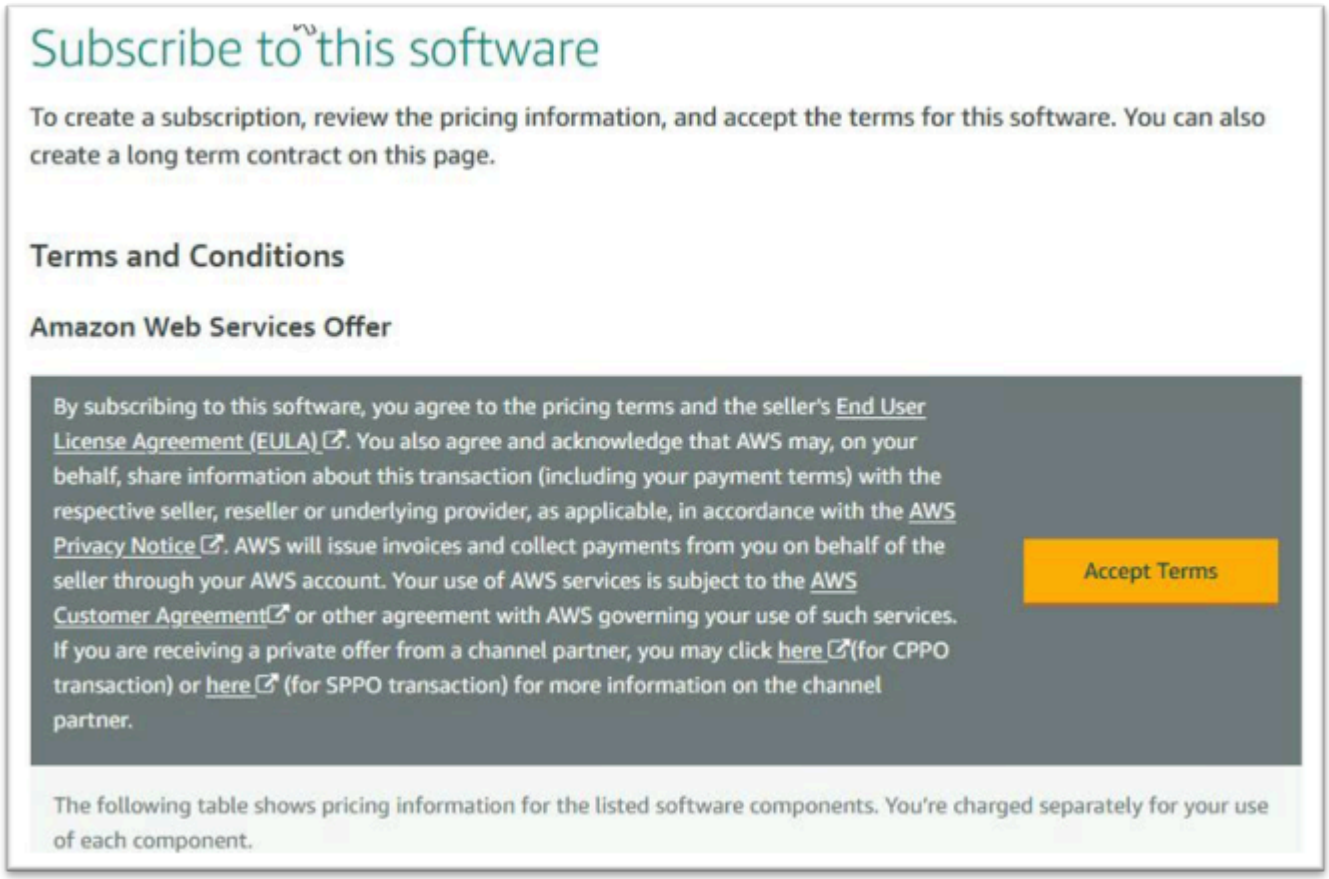
3. 將下列其中一個連結複製並貼到瀏覽器網址列中。

> **Note**
>
> 僅選擇您已獲授權使用的產品之一的鏈接。

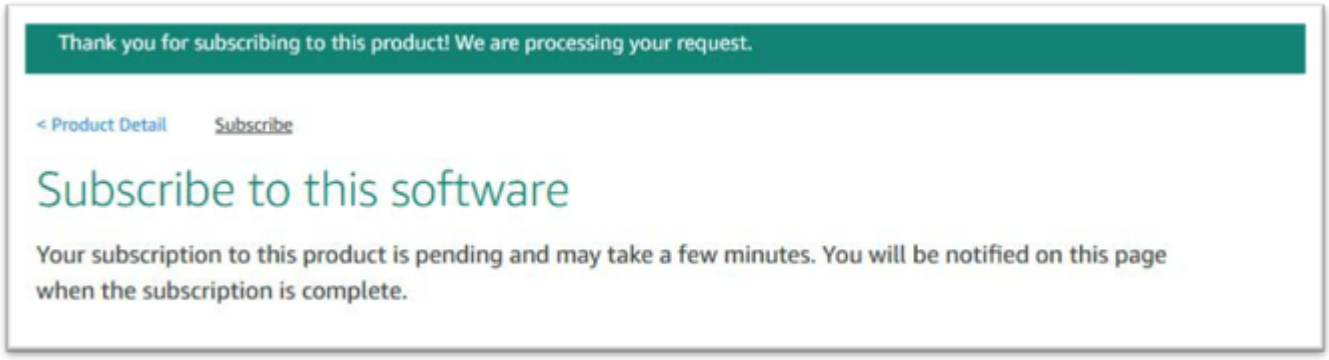
- 企業伺服器：https://aws.amazon.com/marketplace/PP /產品-g5ev6L7
- 企業服務器的窗口：https://aws.amazon.com/marketplace/pp/產品視窗
- 企業開發人員：https://aws.amazon.com/marketplace/PP /產品-77qr42WK
- 具有視覺工作室 2022 的企業開發人員：https://aws.amazon.com/marketplace/pp/產品-M4L3
- 企業級分析儀：https://aws.amazon.com/marketplace/PP/產品展示
- 企業構建工具的視窗:https://aws.amazon.com/marketplace/pp/產品-2rw35 英寸
- 企業預存程序：https://aws.amazon.com/marketplace/pp/產品預存程序
- 使用SQL伺服器 2019 年的企業預存程序：https://aws.amazon.com/marketplace/pp/產品預存程序 4

4. 選擇 Continue to Subscribe (繼續以訂閱)。

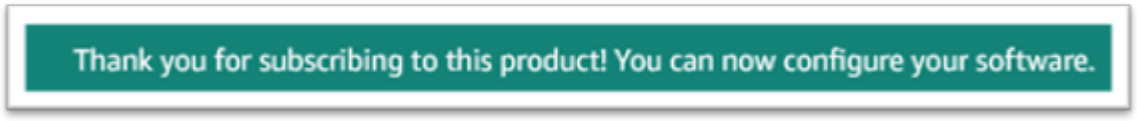
5. 如果「條款與條件」可接受，請選擇「接受條款」。



6. 訂閱可能需要幾分鐘的時間來處理。

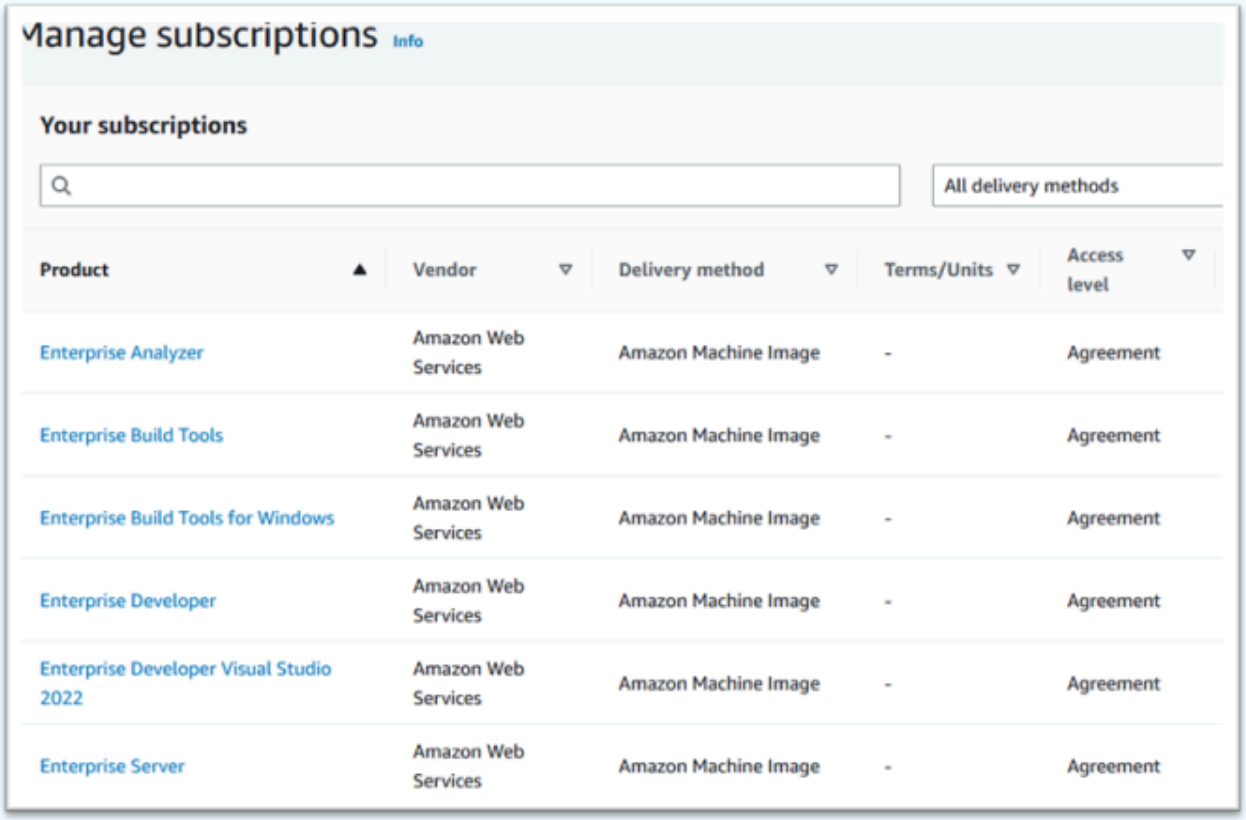


7. 顯示感謝訊息後，複製並貼上步驟 3 中的下一個連結，以繼續新增訂閱。



8. 當「管理訂閱」顯示所有訂閱項目時停止AMIs。

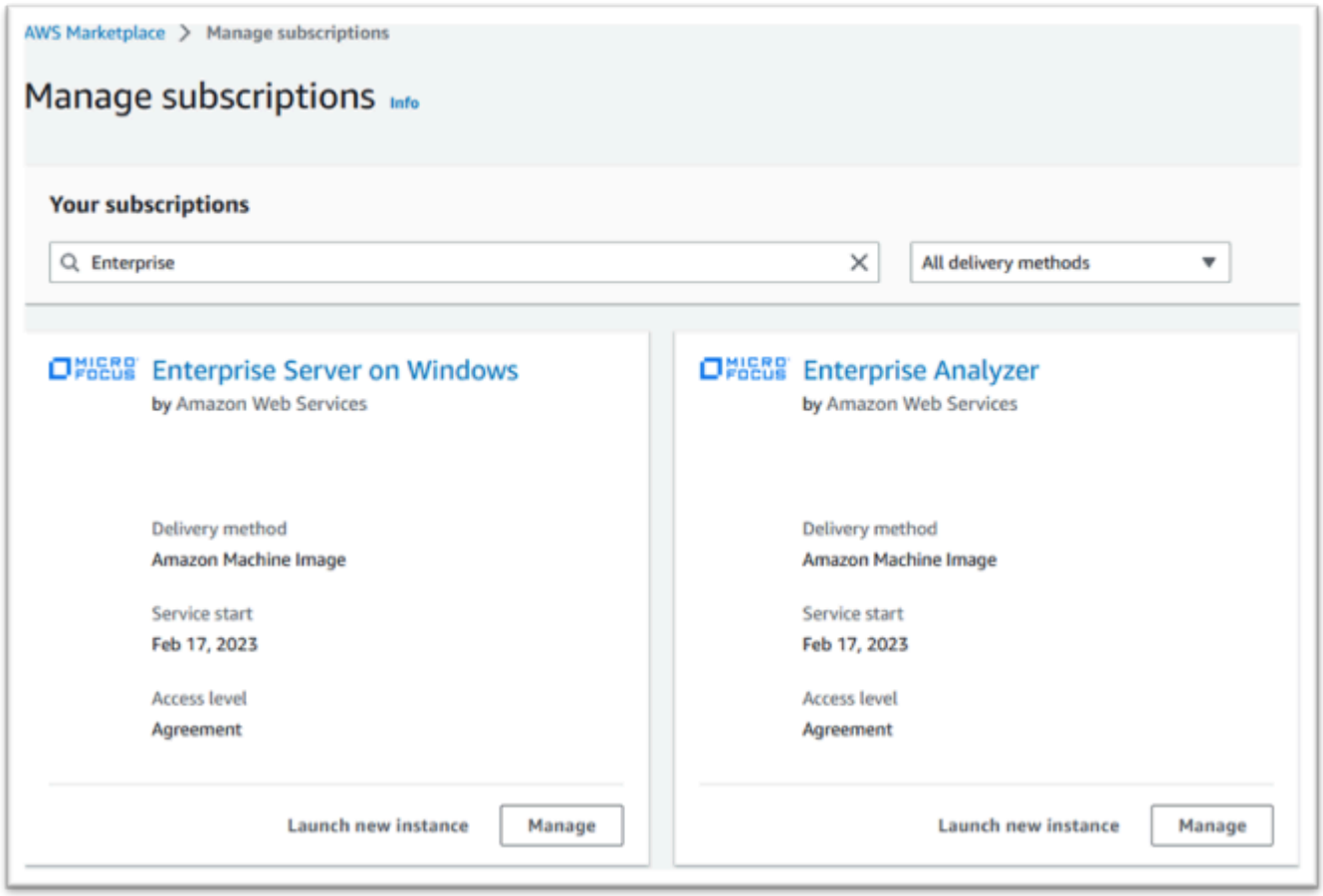
> **Note**
>
> 面板偏好設定 (齒輪圖示) 設定為將「檢視」顯示為「表格」。



啟動 AWS Mainframe Modernization 微焦點實體

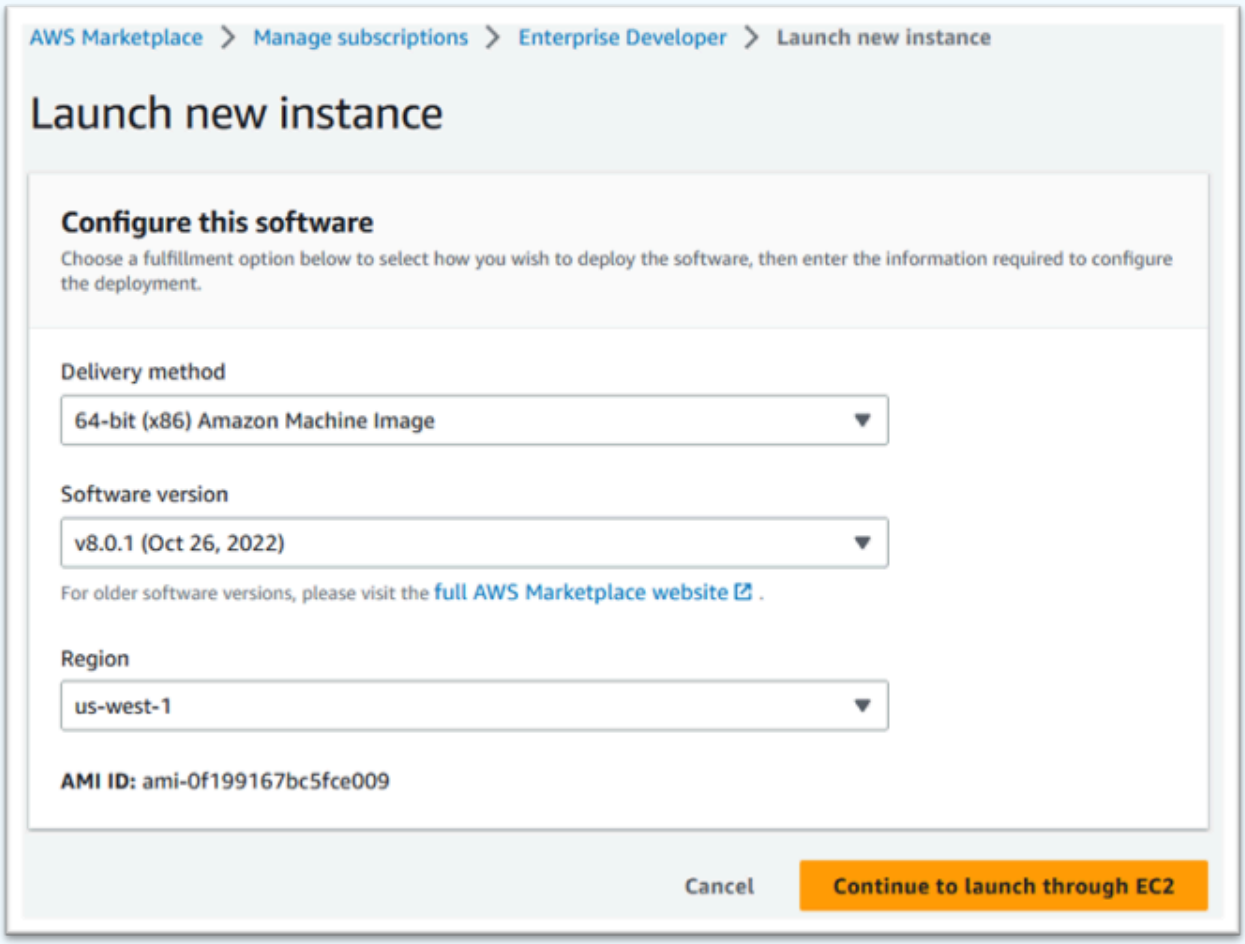
建立端點、IAM策略、IAM角色和訂閱之後AMIs，您就可以在中啟動 AWS Mainframe Modernization Micro Focus 執行個體 AWS Management Console。

1. 導覽至中的「 AWS Marketplace 訂閱」 AWS Management Console。
2. 找到AMI要啟動的，然後選擇啟動新實例。



3. 在啟動新執行個體對話方塊中，確定已選取允許列出的區域。
4. 按「繼續」以啟動EC2。

> **Note**
>
> 下面的例子顯示了一個企業開發人員的啟動AMI，但這個過程是相同的所有 AWS Mainframe Modernization AMIs。

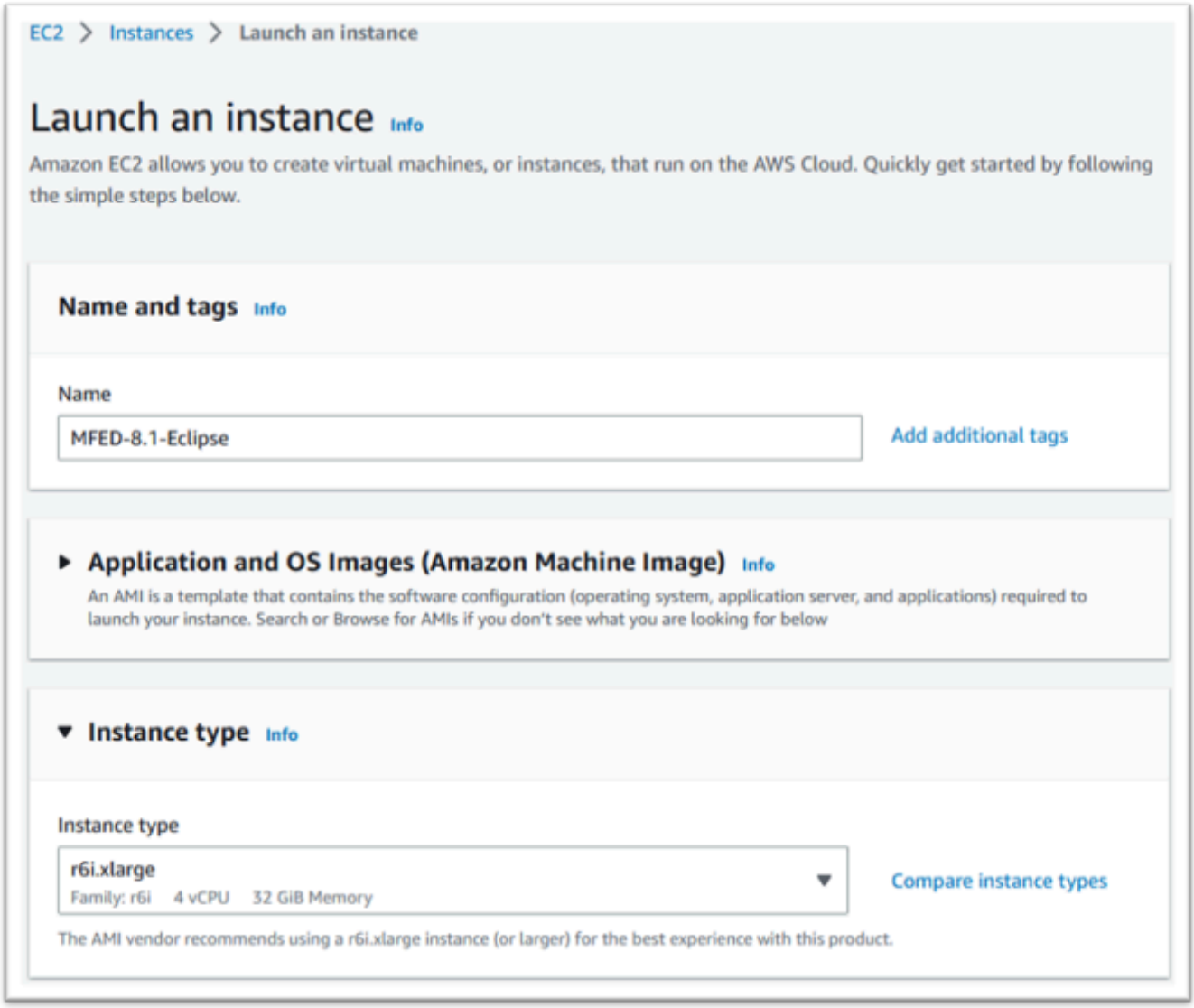


5. 輸入伺服器的名稱。
6. 選擇執行個體類型。

   選取的執行個體類型應由專案效能和成本需求決定。以下是建議的起點：

   - 對於企業分析儀，一個 r6i.xlarge
   - 對於企業開發人員，一個 r6i.large
   - 如果是獨立的企業伺服器執行個體，則為 r6i.xlarge
   - 對於具有向外擴充的微焦點效能可用性叢集 (PAC)，則為 r6i.large

   > **Note**
   >
   > 螢幕擷取畫面的「應用程式和作業系統映像」區段已收合。



7. 選擇或建立 (並儲存) 金鑰配對 (未顯示)。

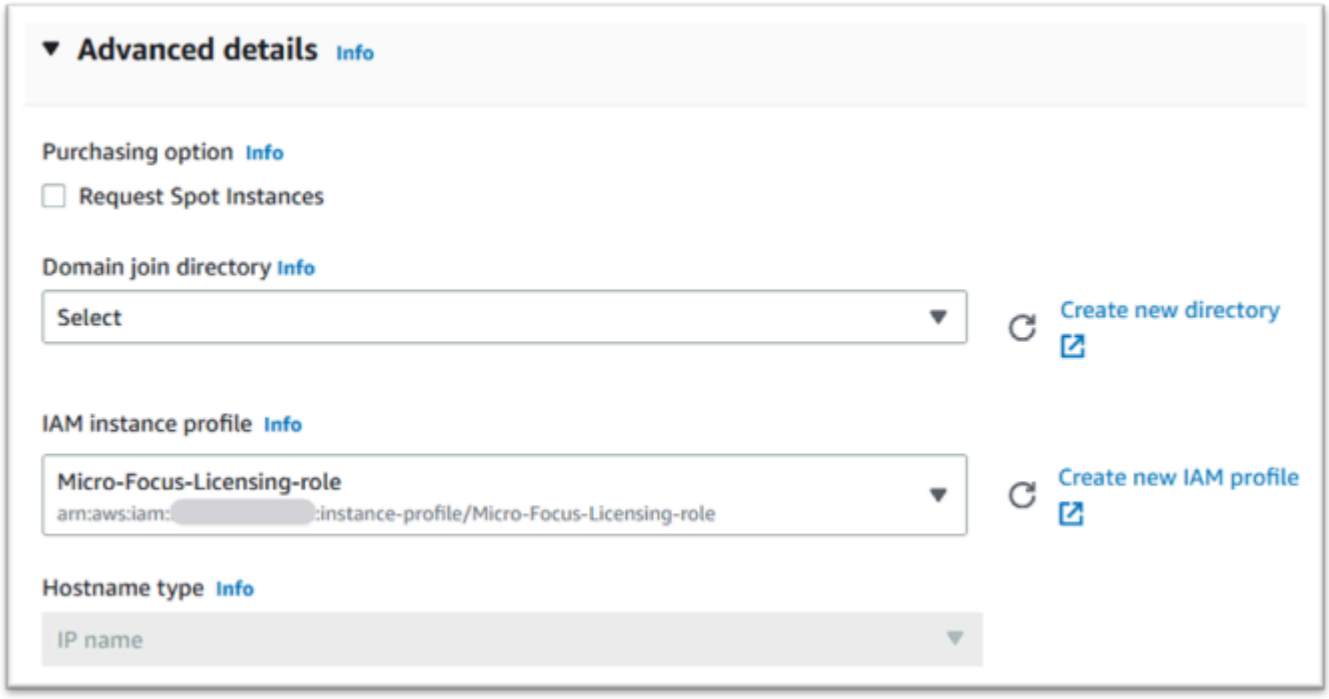
   如需 Linux 執行個體金鑰配對的詳細資訊，請參閱 Amazon EC2 金鑰配對和 Linux 執行個體。

   如需 Windows 執行個體金鑰配對的詳細資訊，請參閱 Amazon EC2 金鑰配對和 Windows 執行個體。

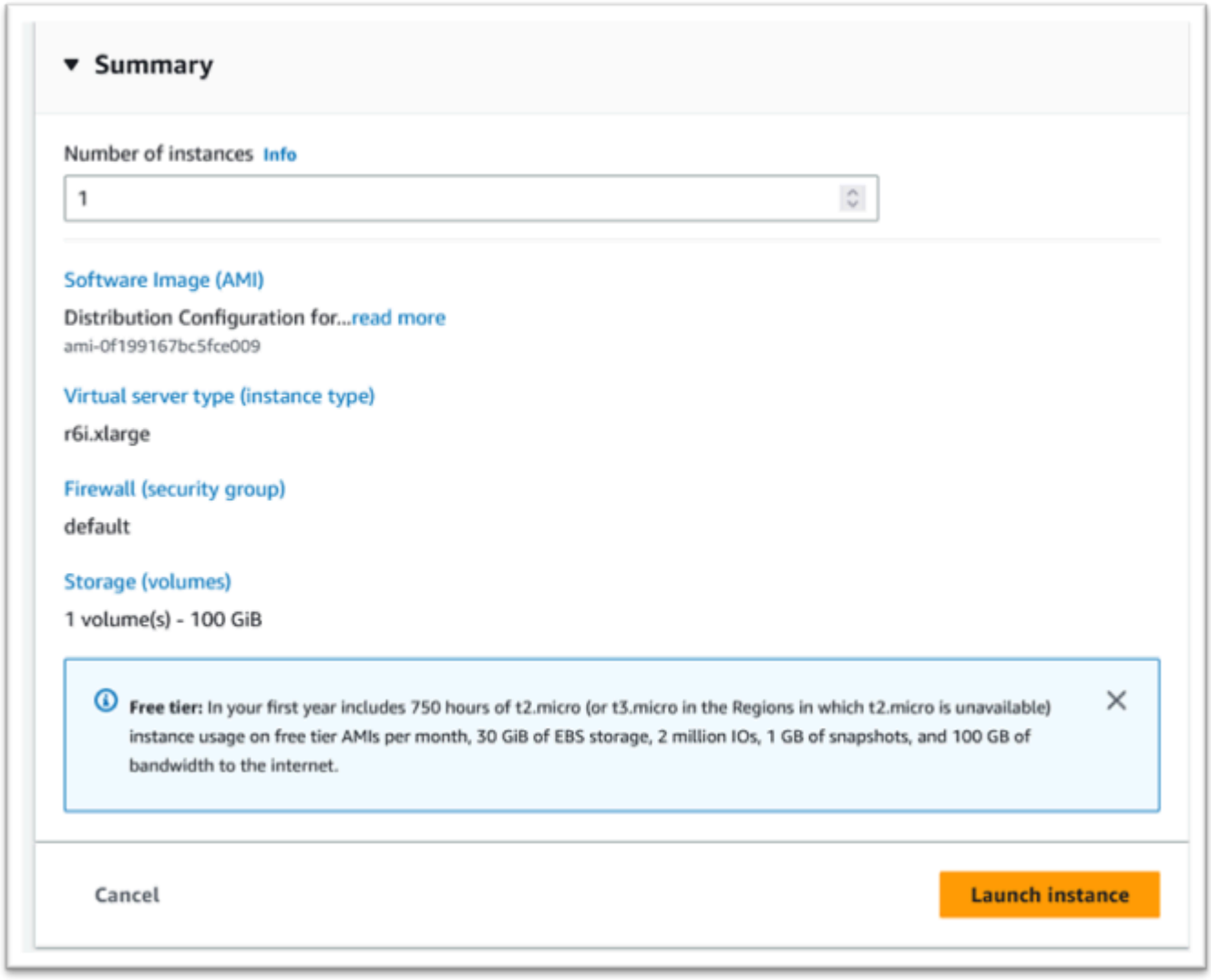
8. 編輯網路設定，然後選擇允許列出VPC和適當的子網路。
9. 選擇或建立「安全性群組」。如果這是企業伺服器EC2執行個體，TCP通常會允許連接埠 86 和 10086 的流量來管理 Micro Focus 組態。
10. 選擇性地設定 Amazon EC2 執行個體的儲存。

11. 重要-展開 [進階詳細資料]，然後在IAM執行個體設定檔下選擇先前建立的授權角色，例如「微焦點-授權-角色」。

> **Note**
>
> 如果遺漏此步驟，建立執行個體之後，您可以從EC2執行個體的 [動作] 功能表的 [安全性] 選項修改IAM角色。

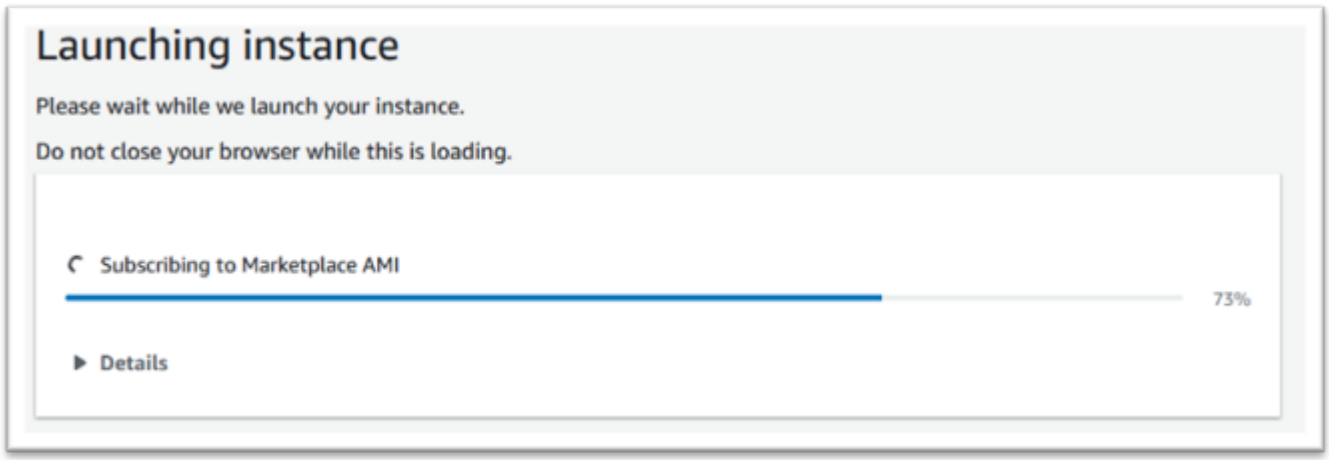


12. 複查「彙總」並推入啟動執行環境。

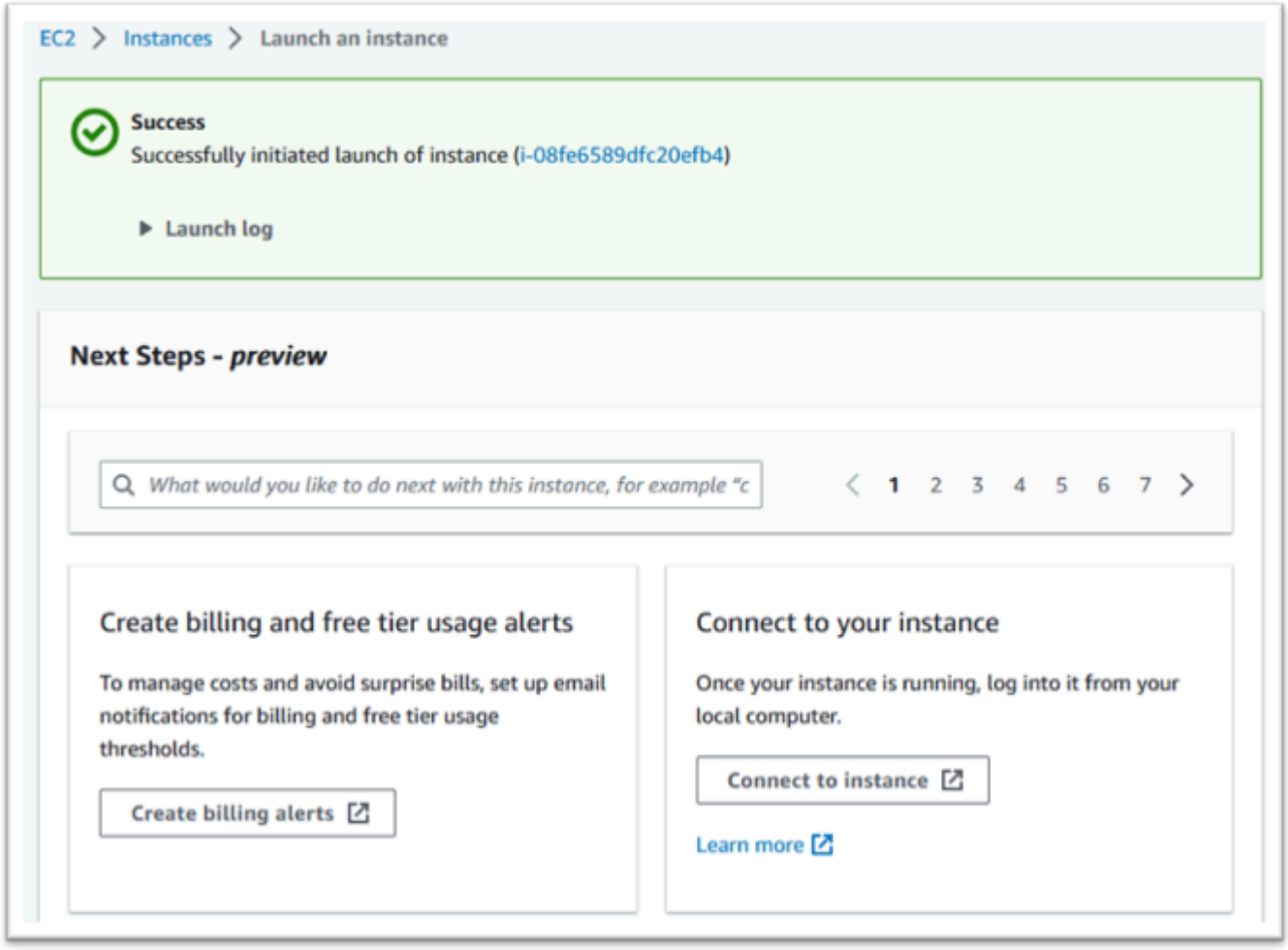


13. 如果選擇了無效的虛擬伺服器類型，執行個體啟動將會失敗。

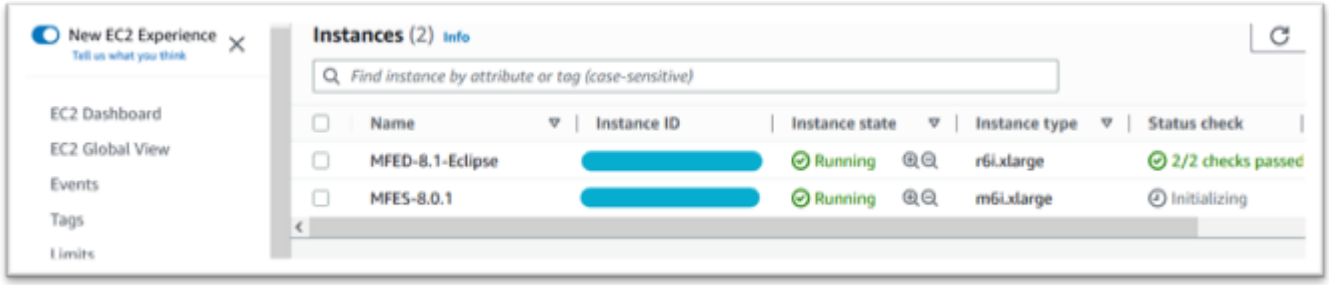
    如果發生這種情況，請選擇編輯實例配置並更改實例類型。



14. 顯示「成功」訊息後，選擇 [Connect 線至執行個體] 以取得連線詳細資料。



15. 或者，導覽至EC2中的 AWS Management Console。
16. 選擇「執行個體」以查看新執行個體的狀態。



子網路或沒VPC有網際網路存取

如果子網路或VPC沒有輸出網際網路存取權，請進行這些額外的變更。

授權管理員需要存取下列AWS服務：

- COM. 亞馬遜。*region*.S3
- COM. 亞馬遜。*region*.ec2
- COM. 亞馬遜。*region*. 許可證管理器
- COM. 亞馬遜。*region*. STS

前面的步驟定義了。*region*.s3 服務作為閘道端點。對於沒有網際網路存取權的任何子網路，此端點需要一個路由表項目。

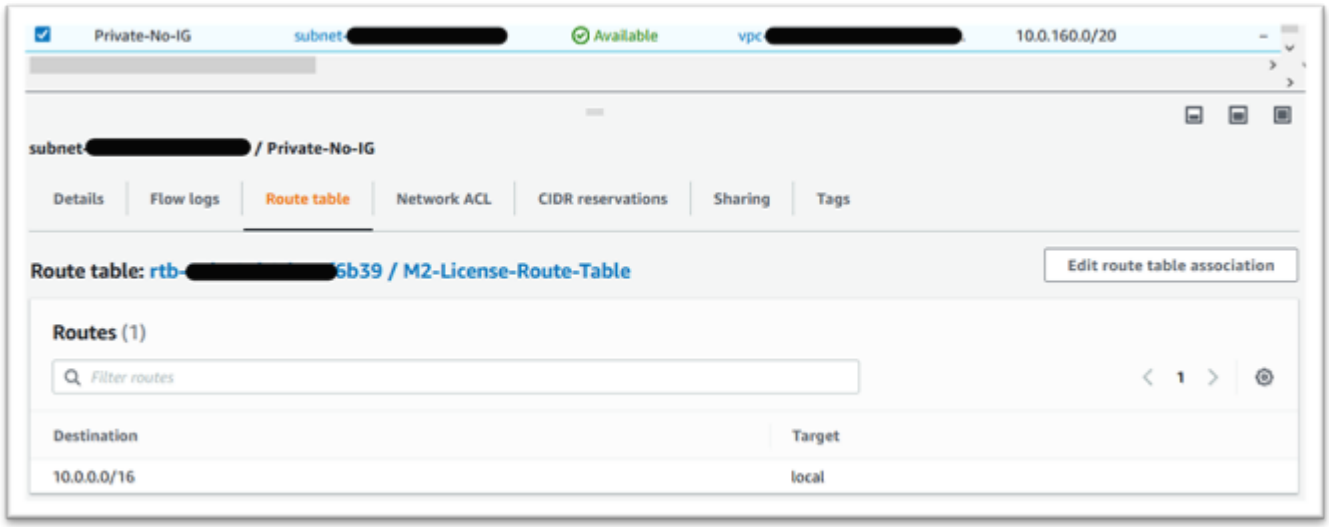
額外的三個服務將被定義為接口端點。

主題

- 新增 Amazon S3 端點的路由表項目
- 定義所需的安全性群組
- 建立服務端點

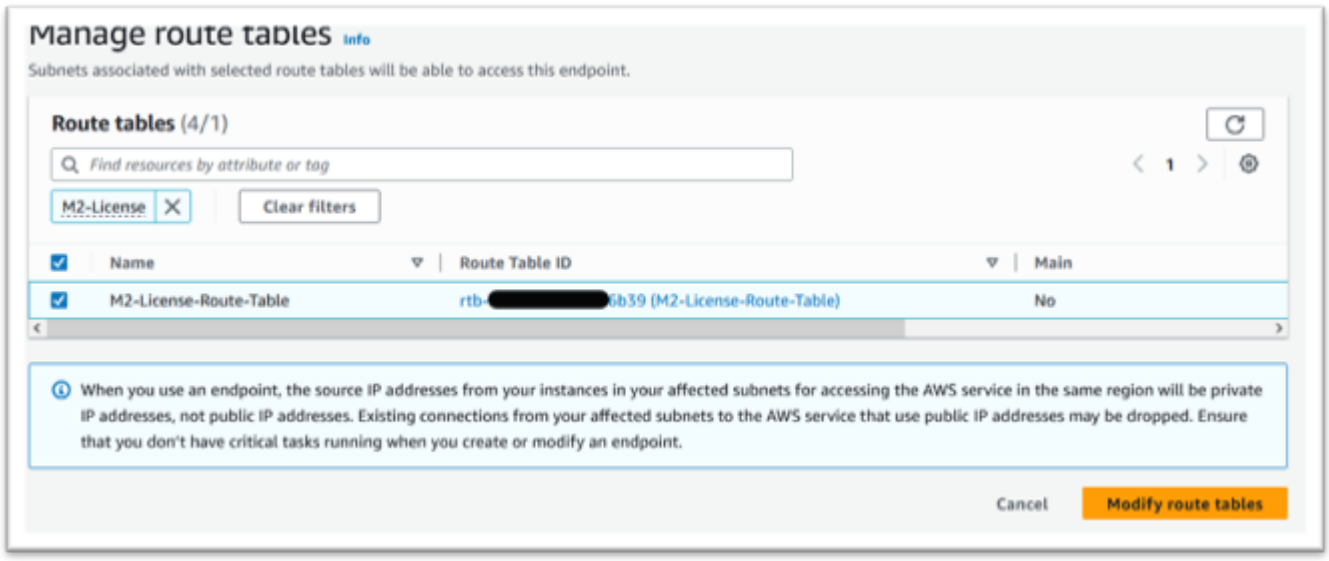
新增 Amazon S3 端點的路由表項目

1. 導覽至VPC中， AWS Management Console 然後選擇「子網路」。
2. 選擇要在其中建立 Amazon EC2 執行個體的子網路，然後選擇「路由表」索引標籤。
3. 請注意路由表 ID 的幾個尾隨數字。例如，下圖中的 6b39。



4. 從導覽窗格中選擇「端點」。

5. 從端點的「路由表」索引標籤，或從「動作」下拉式清單中選擇先前建立的端點，然後選擇「管理路由表」表。
6. 使用前面識別的數字選擇路由表，然後按修改路由表。

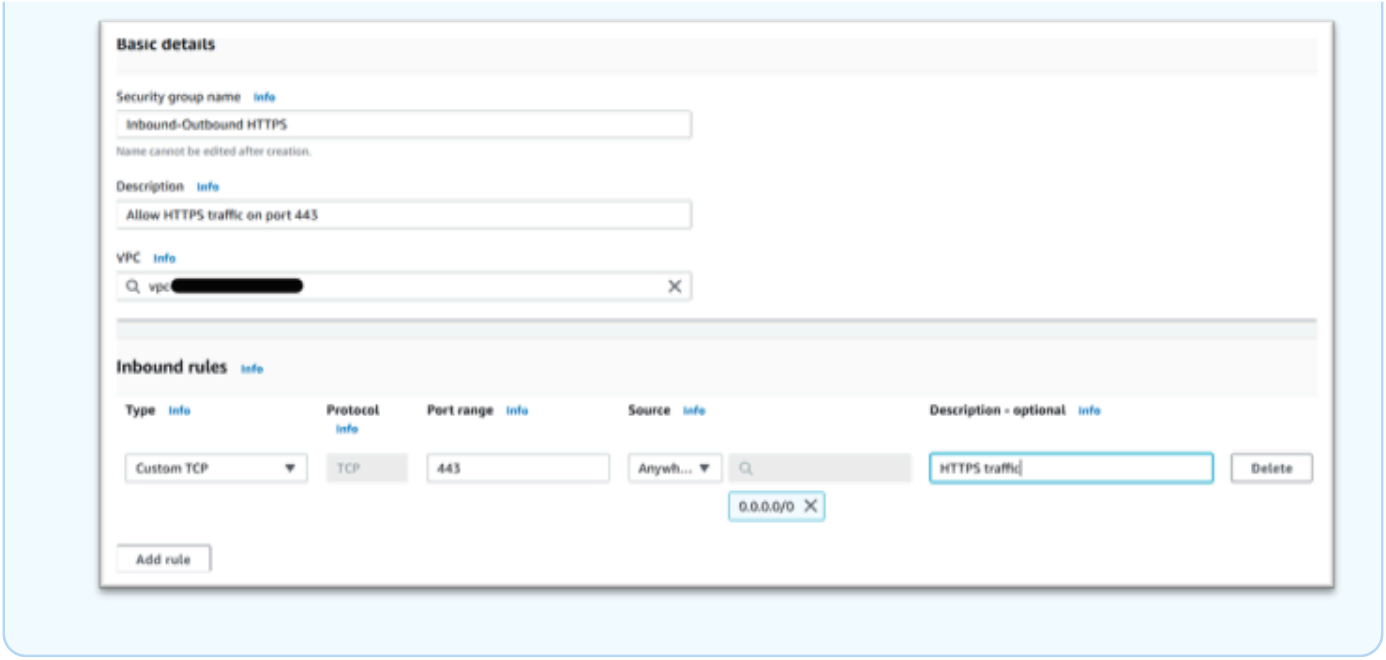


定義所需的安全性群組

Amazon EC2 AWS STS、和 License Manager 服務HTTPS透過連接埠 443 進行通訊。此通訊為雙向通訊，需要輸入和輸出規則，才能允許執行個體與服務通訊。

1. 導航到 Amazon VPC 在 AWS Management Console.
2. 在導覽列中找到「安全群組」，然後選擇「建立安全性群組」
3. 輸入「安全性」群組名稱和說明，例如「入站-輸出HTTPS」。
4. 按下VPC選取區域中的 X 以移除預設值 VPC，然後選擇包VPC含 S3 端點的端點。
5. 新增輸入規則，以允許來自任何位置連接埠 443 的TCP流量。

> **Note**
>
> 入站（和出站規則）可以通過限制源進一步限制。如需詳細資訊，請參閱 Amazon 使用VPC者指南中的使用安全群組控制 AWS 資源流量。

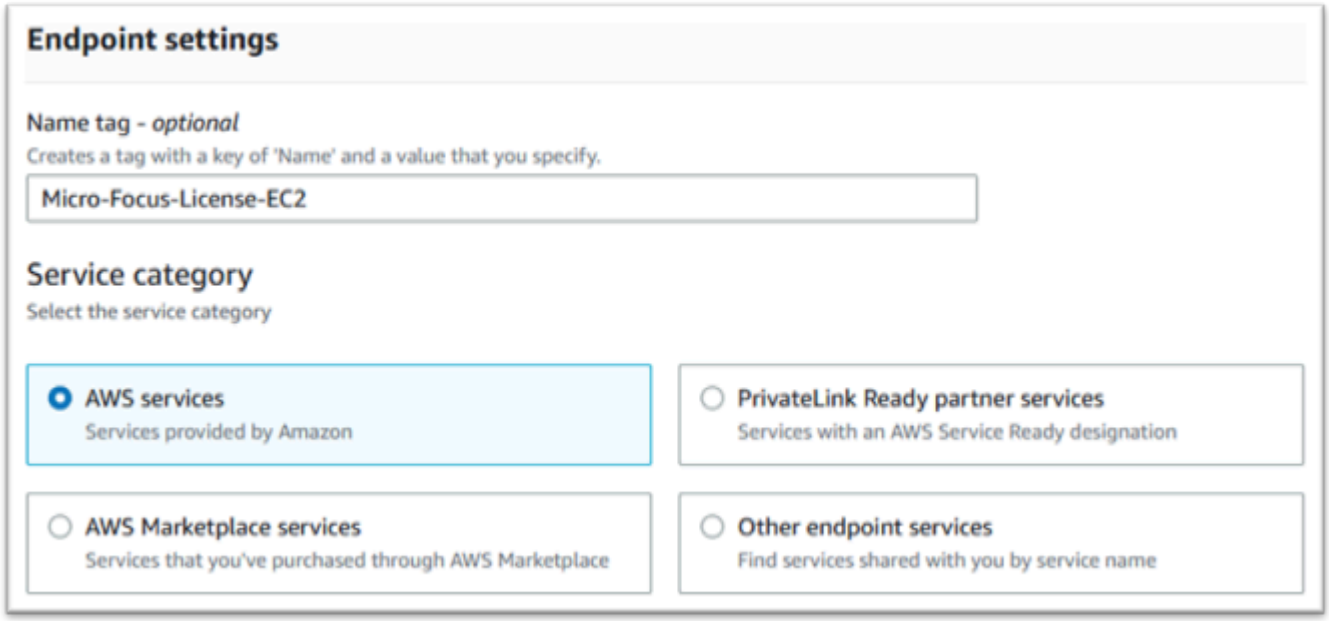


6. 按創建安全組。

建立服務端點

重複此程序三次，每項服務一次。

1. VPC在中導航到 Amazon，然 AWS Management Console 後選擇端點。
2. 按下「建立端點」。
3. 輸入名稱，例如「微焦點授權-」、「微焦點授權-EC2」或「微焦點-授權管理員STS」。
4. 選擇「AWS服務服務」類別。



5. 在「服務」下搜索匹配的接口服務，該服務是下列其中一個：

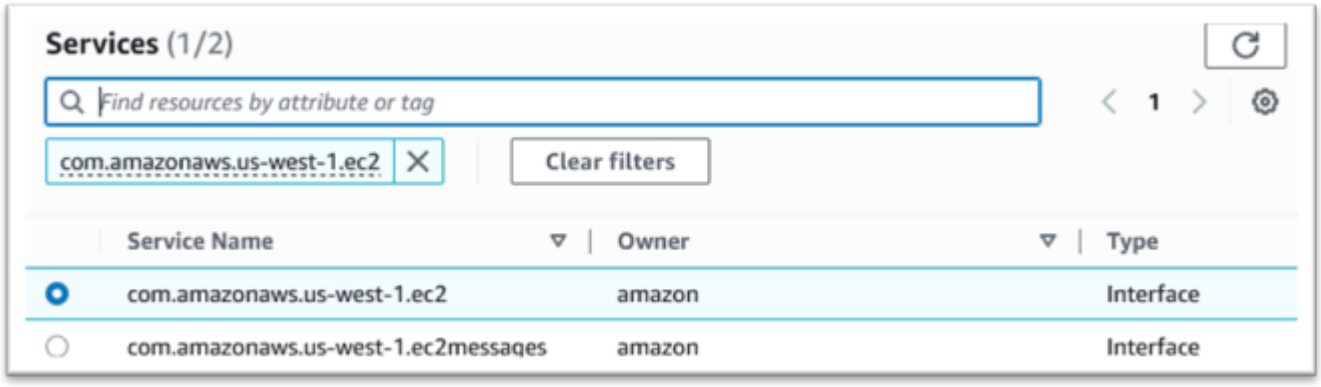
   - 「喜歡。*region*.ec2」
   - 「喜歡。*region*. STS」
   - 「喜歡。*region*. 授權管理員」

   例如：

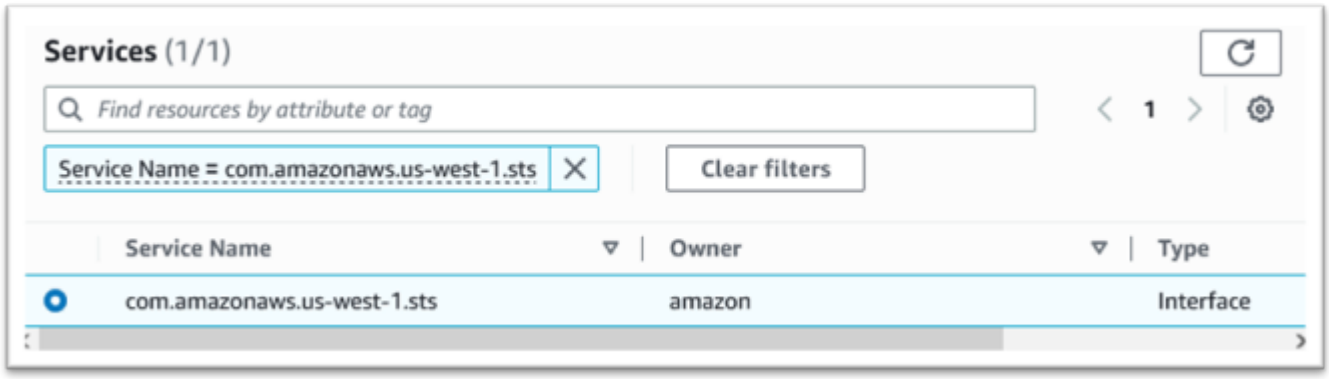
   - 「亞馬遜美國西部 1.ec2」
   - 「我們西部-1. 阿馬遜」
   - 「我們西部-1. 許可證管理器」

6. 選擇匹配的接口服務。

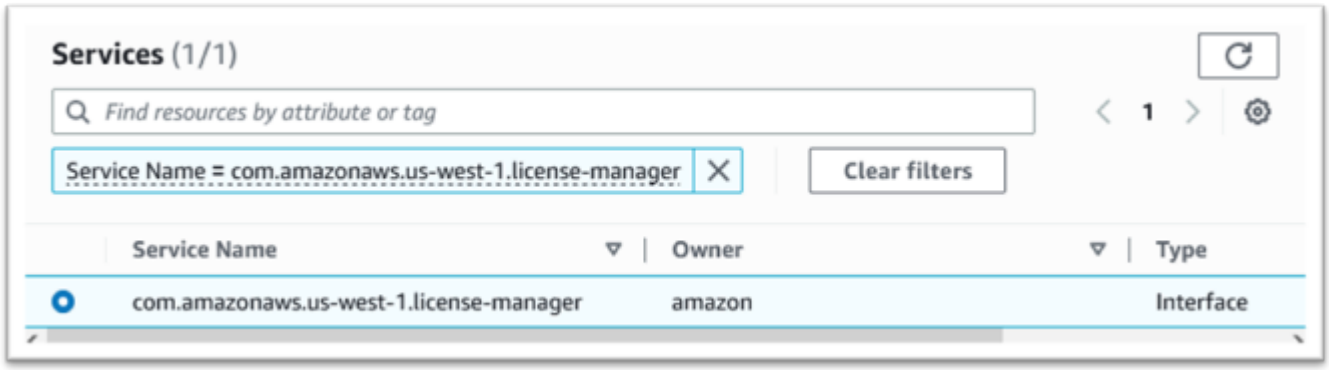
   COM. 亞馬遜。***region***.ec2：



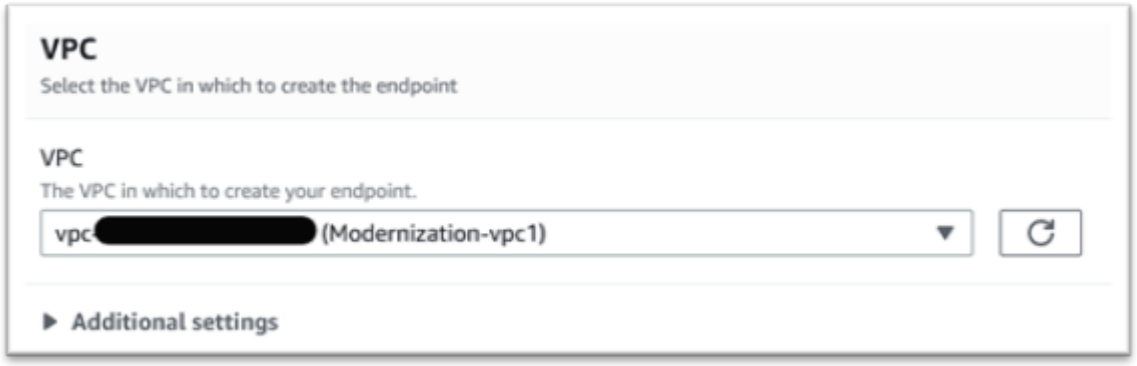
COM. 亞馬遜。***region***. STS：



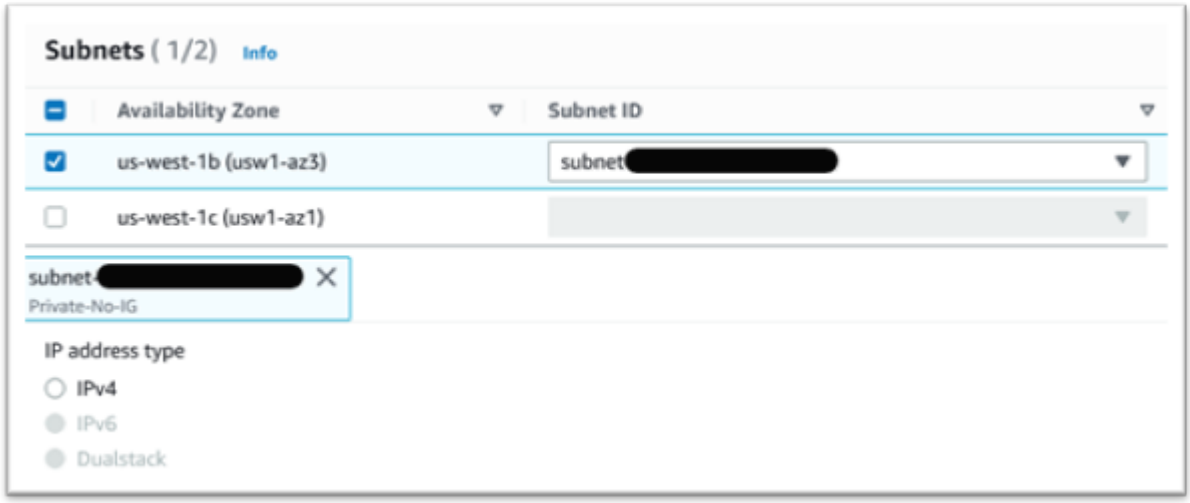
COM. 亞馬遜。***region***. 授權管理員：



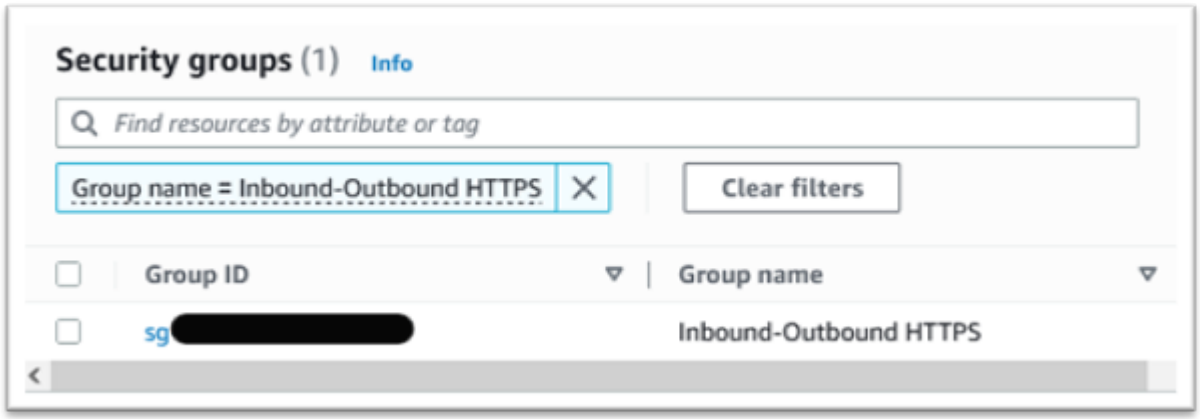
7. 對於VPC選VPC擇實例。



8. 選擇的可用區域和子網路。VPC



9. 選擇之前創建的安全組。



10. 在「策略」下選擇「完整存取

11. 選擇建立端點。
12. 對其餘介面重複此程序。

為 Micro Focus 企業分析儀和 Micro Focus 企業開發人員串流會議設定自動化

您可以在工作階段開始和結束時自動執行指令碼，以允許特定於客戶內容的自動化。如需此 AppStream 2.0 功能的詳細資訊，請參閱《Amazon 2.0 管理指南》中的 AppStream 使用工作階段指令碼管理 AppStream 2.0 使用者的串流體驗。

此功能需要您至少具有下列版本的企業分析器和企業開發人員映像檔：

- `m2-enterprise-analyzer-v8.0.4.R1`
- `m2-enterprise-developer-v8.0.4.R1`

主題

- 在工作階段開始時設定自動化
- 在工作階段結束時設定自動化

在工作階段開始時設定自動化

如果您想要在使用者連線到 AppStream 2.0 時執行自動化指令碼，請建立您的指令碼並命名`m2-user-setup.cmd`。將腳本存儲在用戶的 AppStream 2.0 主文件夾中。 AWS 大型主機現代化提供的 AppStream 2.0 映像會在該位置尋找具有該名稱的指令碼，並執行它 (如果存在)。

> **Note**
>
> 指令碼持續時間不能超過 AppStream 2.0 設定的限制，目前為 60 秒。如需詳細資訊，請參閱 Amazon AppStream 2.0 管理指南中的串流工作階段開始之前執行指令碼。

在工作階段結束時設定自動化

如果您想要在使用者從 AppStream 2.0 中斷連線時執行自動化指令碼，請建立您的指令碼並命名`m2-user-teardown.cmd`。將腳本存儲在用戶的 AppStream 2.0 主文件夾中。 AWS 大型主機現代化提供的 AppStream 2.0 映像會在該位置尋找具有該名稱的指令碼，並執行它 (如果存在)。

> **Note**
>
> 指令碼持續時間不能超過 AppStream 2.0 設定的限制，目前為 60 秒。如需詳細資訊，請參閱 Amazon AppStream 2.0 管理指南中的串流工作階段結束後執行指令碼。

在企業開發人員中以表格和欄的形式檢視資料集

您可以使用 Micro Focus 執行階段存取在大型主機現代化中部署的 AWS 大型主機資料集。您可以從 Micro Focus 企業開發人員執行個體以表格和欄的形式檢視移轉的資料集。以這種方式檢視資料集可讓您：

- 對移轉的`SQL SELECT`資料檔執行作業。
- 在移轉的大型主機應用程式之外公開資料，而不需變更應用程式。
- 輕鬆過濾數據並另存為CSV或其他文件格式。

> **Note**
>
> 步驟 1 和 2 是一次性活動。為每個資料集重複步驟 3 和 4，以建立資料庫檢視。

主題

- 必要條件
- 步驟 1：設定與 Micro Focus 資料存放區的ODBC連線 (Amazon 資RDS料庫)

- 步驟 2：建立 MFDBFH .cfg 檔案
- 步驟 3：為您的字帖版面建立結構 (STR) 檔案
- 第 4 步：使用結構（STR）文件創建數據庫視圖
- 步驟 5：以表格和欄的形式檢視微型焦點資料集

必要條件

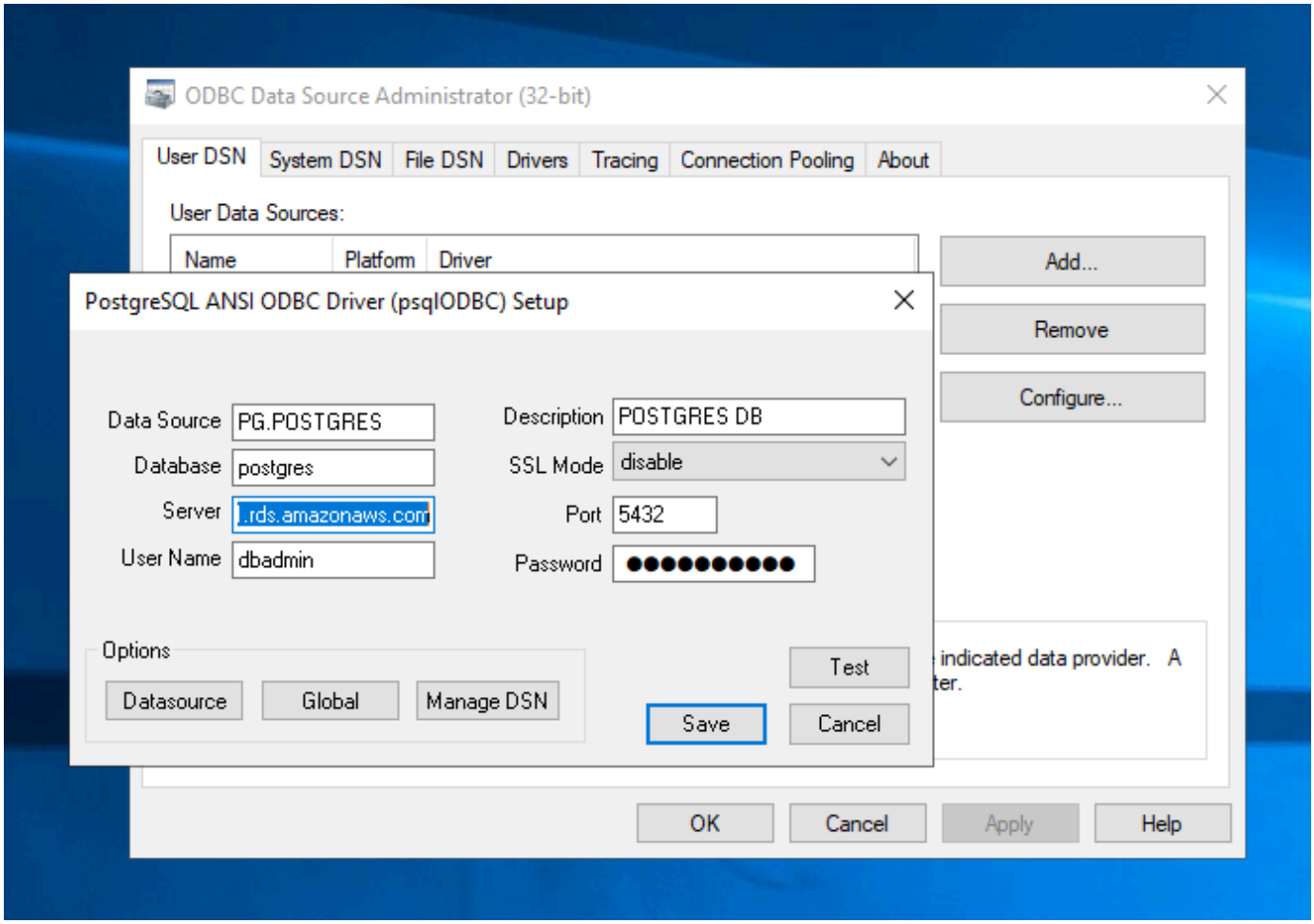
- 您必須能夠透過 AppStream 2.0 存取 Micro Focus 企業開發人員桌面。
- 您必須使用 Micro Focus 執行階段引擎，在 AWS 大型主機現代化下部署並執行應用程式。
- 您正在將應用程序數據存儲在 Aurora Postgre SQL 兼容版本中。

步驟 1：設定與 Micro Focus 資料存放區的ODBC連線 (Amazon 資RDS料庫)

在此步驟中，您會設定資料庫的ODBC連線，該連線包含您要以資料表和欄的形式檢視的資料。這只是一次性的步驟。

1. 使用 AppStream 2.0 串流登入 Micro Focus 企業開發者桌面版URL。
2. 開啟ODBC資料來源管理員，選擇 [使用者]，DSN然後選擇 [新增
3. 在「建立新資料來源」中，選擇「下一步」，SQLANSI然後選擇「完成」。
4. PG.POSTGRES透過提供必要的資料庫資訊來建立資料來源，如下所示：

```
Data Source : PG.POSTGRES
Database    : postgres
Server      : rds_endpoint.rds.amazonaws.com
Port        : 5432
User Name   : user_name
Password    : user_password
```



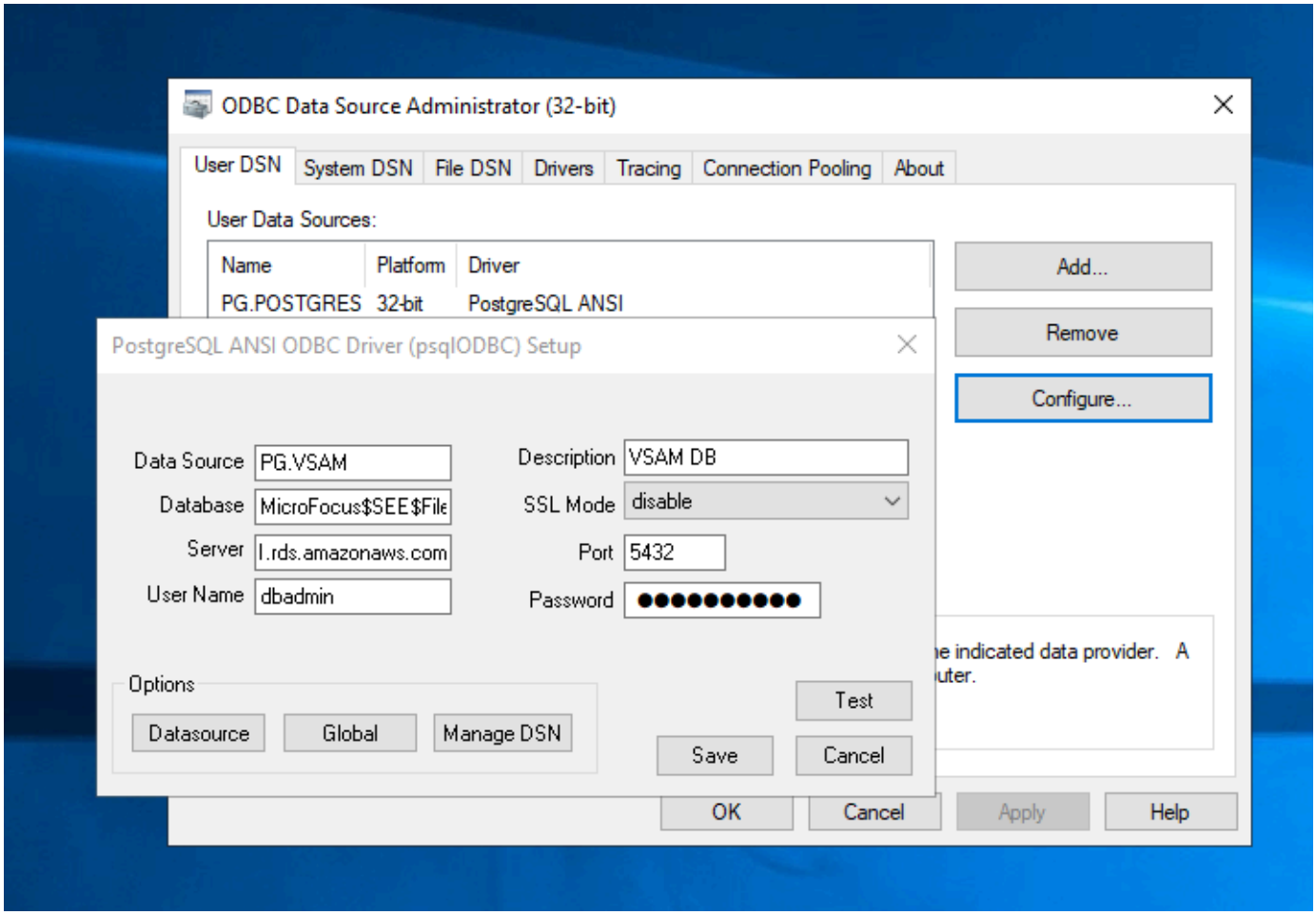
5. 選擇「測試」以確保連接正常。`Connection successful`如果測試成功，您應該會看到該消息。

   如果測試未成功，請檢閱下列資訊。

   - Amazon 故障排除 RDS
   - 如何解決連線到 Amazon RDS 資料庫執行個體時的問題？

6. 儲存資料來源。
7. 建立資料來源`PG.VSAM`、測試連線並儲存資料來源。提供下列資料庫資訊：

```
Data Source : PG.VSAM
Database    : MicroFocus$SEE$Files$VSAM
Server      : rds_endpoint.rds.amazonaws.com
Port        : 5432
User Name   : user_name
Password    : user_password
```



步驟 2：建立 MFDBFH .cfg 檔案

在此步驟中，您會建立描述 Micro Focus 資料存放區的組態檔案。這是一次性的設定步驟。

1. 例如，在您的主資料夾中`D:\PhotonUser\My Files\Home Folder\MFED\cfg\MFDBFH.cfg`，建立包含下列內容的 MFDBFH .cfg 檔案。

```
<datastores>
       <server name="ESPACDatabase" type="postgresql" access="odbc">
        <dsn name="PG.POSTGRES" type="database" dbname="postgres"/>
        <dsn name="PG.VSAM" type="datastore" dsname="VSAM"/>
       </server>
      </datastores>
```

2. 執行下列命令來查詢 Micro Focus 資料存放區，以確認MFDBFH組態：

```
*###*
*## Test the connection by running the following commands*
*###*

set MFDBFH_CONFIG="D:\PhotonUser\My Files\Home Folder\MFED\cfg\MFDBFH.cfg"

dbfhdeploy list sql://ESPACDatabase/VSAM?folder=/DATA
```

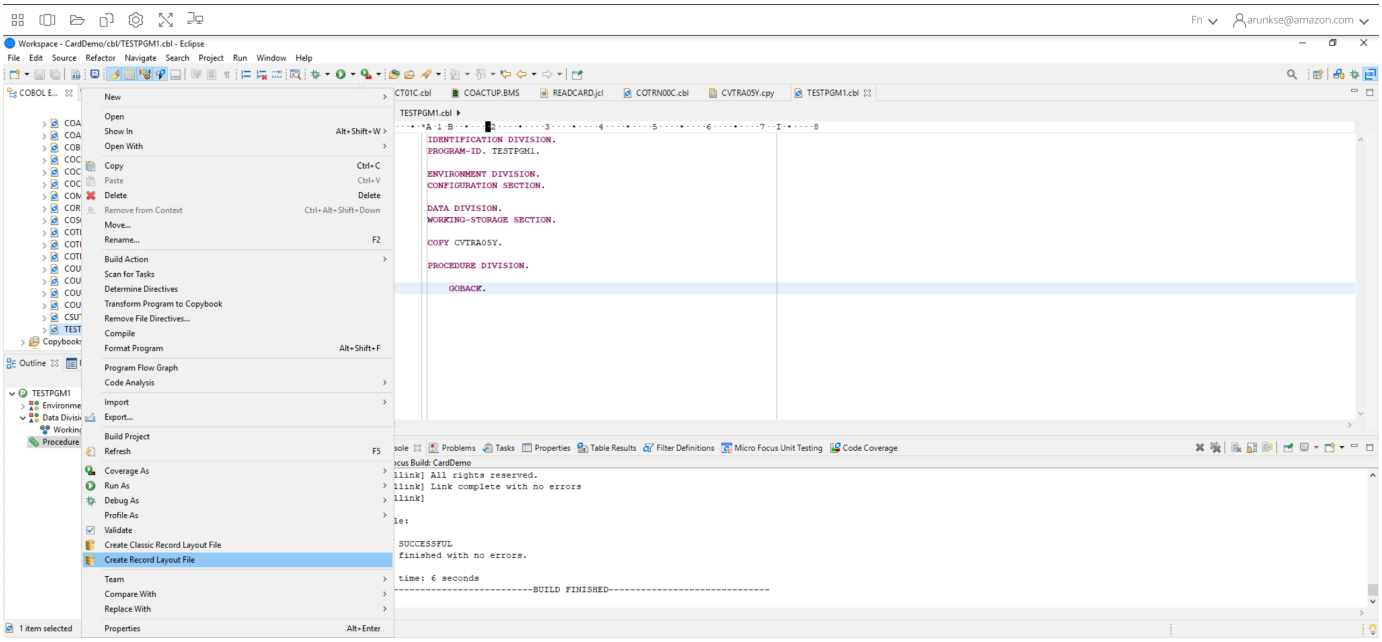
步驟 3：為您的字帖版面建立結構 (STR) 檔案

在此步驟中，您會為您的字稿本版面建立結構檔案，以便稍後使用它從資料集建立資料庫檢視。

1. 編譯與您的字帖相關聯的程序。如果沒有程序正在使用字帖，創建並編譯一個簡單的程序，如下所示，為您的字帖的COPY語句。

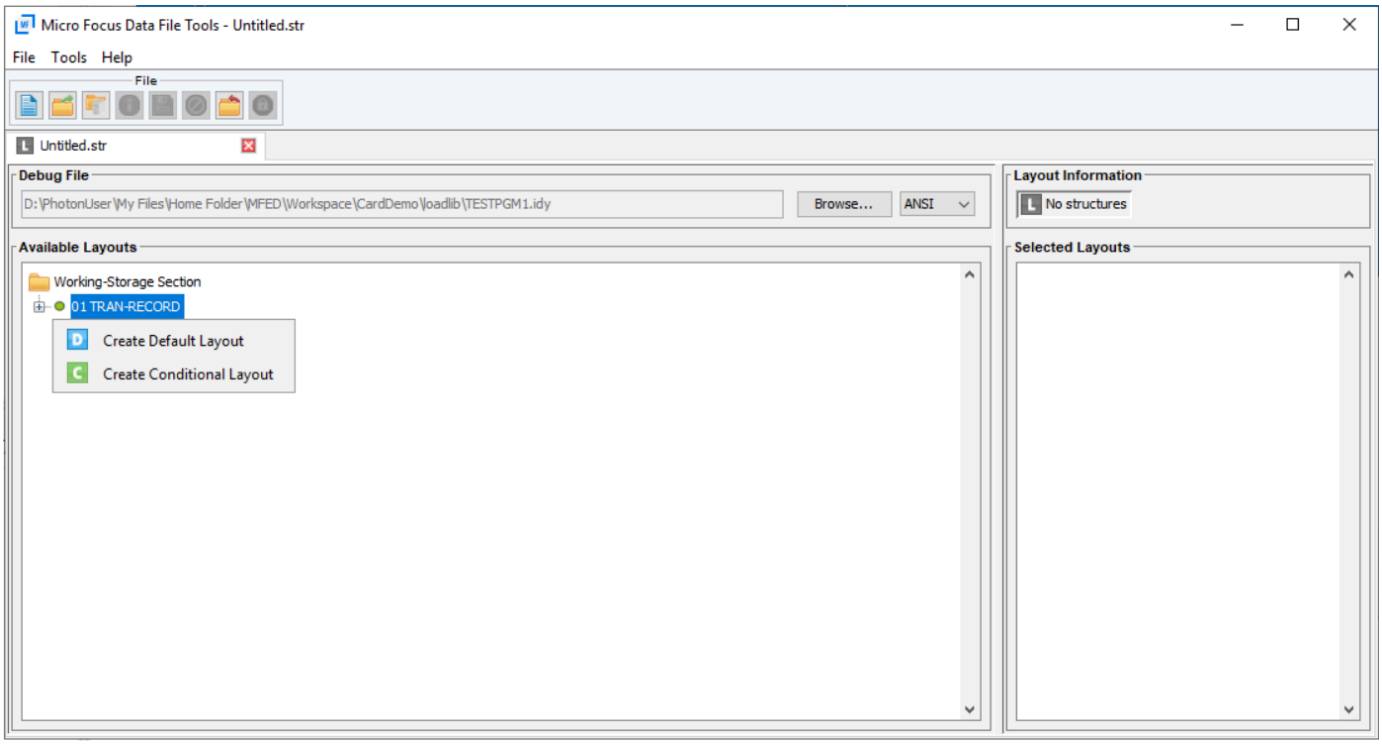
```
IDENTIFICATION DIVISION.
      PROGRAM-ID. TESTPGM1.

      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.

      DATA DIVISION.
      WORKING-STORAGE SECTION.

      COPY CVTRA05Y.

      PROCEDURE DIVISION.

      GOBACK.
```

2. 成功編譯後，右鍵單擊該程序，然後選擇創建記錄佈局文件。這將使用編譯過程中生成的 .idy 文件打開微型焦點數據文件工具。

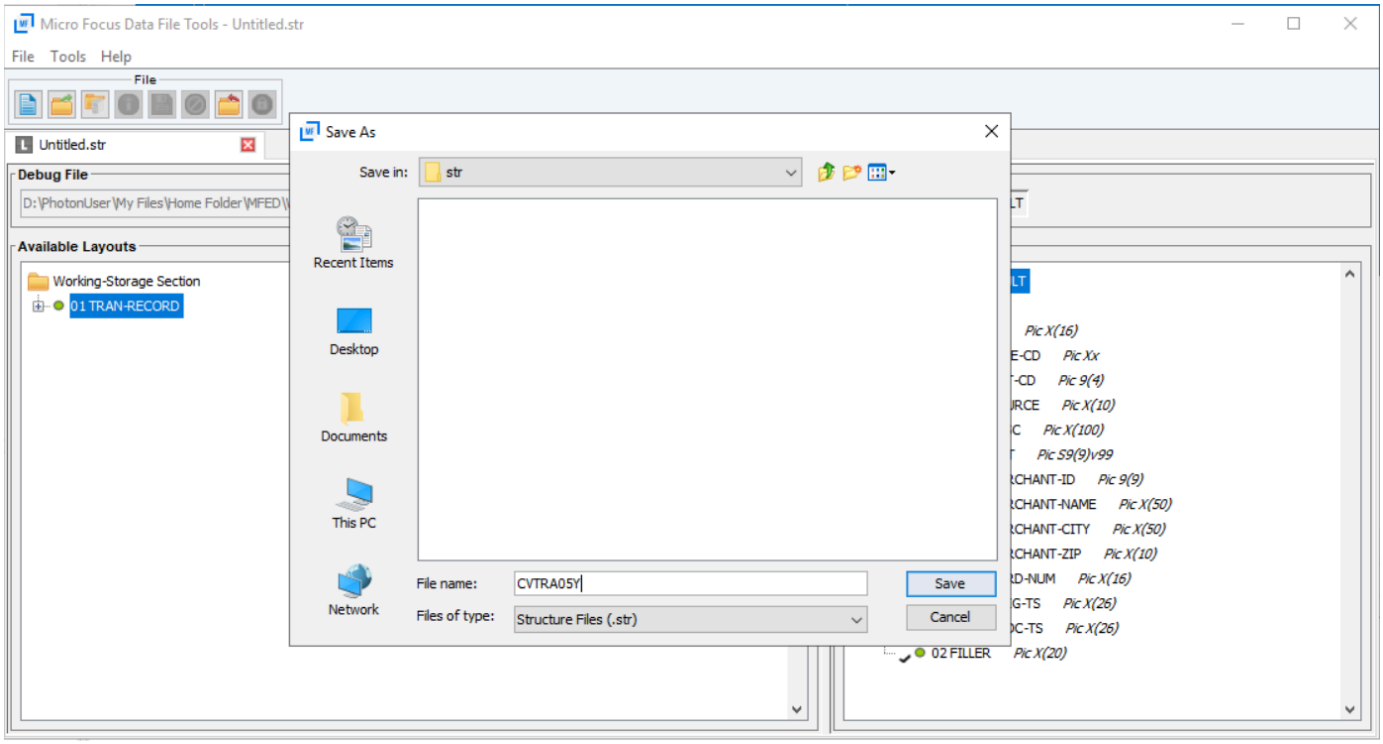


3. 右鍵單擊記錄結構，然後根據佈局選擇創建默認佈局（單個結構）或創建條件佈局（多結構）。

   如需詳細資訊，請參閱 Micro Focus 文件中的建立結構檔案和配置圖。



4. 創建佈局後，從菜單中選擇「文件」，然後選擇「另存為」。瀏覽並將檔案儲存在「主資料夾」下，並使用與您的字帖相同的檔案名稱。您可以選擇創建一個名為的文件夾`str`並將所有結構文件保存在那裡。



第 4 步：使用結構（STR）文件創建數據庫視圖

在此步驟中，您可以使用先前建立的結構檔案來建立資料集的資料庫檢視。

- 使用命dbfhview令為 Micro Focus 資料存放區中已存在的資料集建立資料庫檢視，如下列範例所示。

```
##
      ## The below command creates database view for VSAM file
 AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS
      ## using the STR file CVTRA05Y.str
      ##

      dbfhview -create -struct:"D:\PhotonUser\My Files\Home Folder\MFED\str
\CVTRA05Y.str" -name:V_AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS.DAT -file:sql://
ESPACDatabase/VSAM/AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS.DAT?folder=/DATA

      ##
      ## Output:
      ##

      Micro Focus Database File Handler - View Generation Tool Version 8.0.00
```

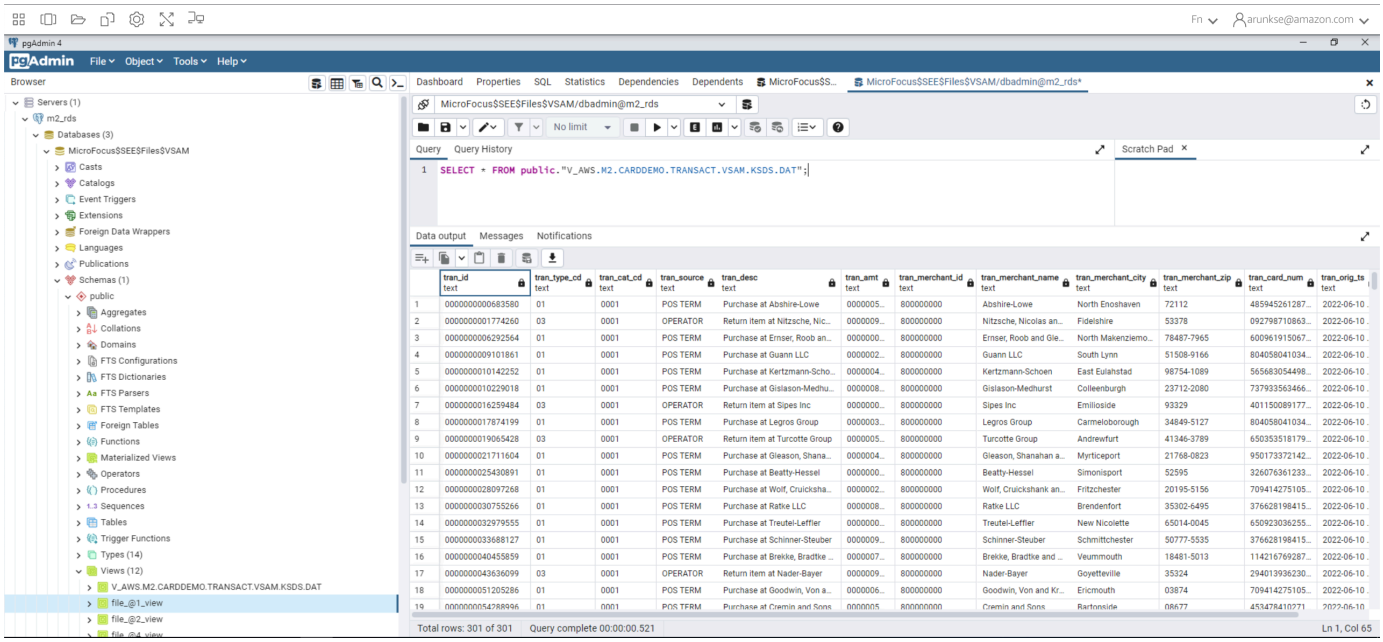
```
      Copyright (C) 1984-2022 Micro Focus. All rights reserved.

      VGN0017I Using structure definition 'TRAN-RECORD-DEFAULT'
      VGN0022I View 'V_AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS.DAT' installed in
datastore 'sql://espacdatabase/VSAM'
      VGN0002I The operation completed successfully
```

步驟 5：以表格和欄的形式檢視微型焦點資料集

在此步驟中，使用連接到數據庫，以pgAdmin便您可以運行查詢以查看數據集，如表和列。

- `MicroFocus$SEE$Files$VSAM`使用 Connect 至資料庫， pgAdmin 並查詢您在步驟 4 中建立的資料庫檢視。

```
SELECT * FROM public."V_AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS.DAT";
```



微焦點教程

本節中的教學課程可協助您開始在 Micro Focus 執行階段引擎中為 AWS 大型主機現代化服務設定各種工作。這些教學課程適用於設定範例應用程式、CI/CD 管線、使用 Micro Focus 企業開發人員的範本，以及設定企業分析器。

主題

- 教學課程：為 BankDemo 範例應用程式設定 Micro Focus 組建
- 教學課程：設定 CI/CD 管道，以便與微焦點企業開發人員搭配使用
- 教學課程：設定 AppStream 2.0 以搭配 Micro Focus 企業分析儀和微焦點企業開發人員使用
- 教學課程：與微焦點企業開發人員搭配使用
- 教學課程：在 AppStream 2.0 上設定企業分析器
- 教學課程：在 AppStream 2.0 上設定微焦點企業開發人員

教學課程：為 BankDemo 範例應用程式設定 Micro Focus 組建

AWS 大型主機現代化讓您能夠為移轉的應用程式設定組建和持續整合/持續交付 (CI/CD) 管線。這些組建和管線會使用 AWS CodeBuild AWS CodeCommit、並 AWS CodePipeline 提供這些功能。CodeBuild 是完全受控的建置服務，可編譯原始程式碼、執行單元測試，以及產生準備好部署的成品。 CodeCommit 是一項版本控制服務，可讓您在雲端中以私密方式儲存和管理 Git 回覆。 AWS CodePipeline 是一項持續交付服務，可讓您建立軟體發行所需步驟的模型、視覺化和自動化。

本教學課程示範如 AWS CodeBuild 何使用從 Amazon S3 編譯 BankDemo 範例應用程式原始碼，然後將編譯後的程式碼匯出回 Amazon S3。

AWS CodeBuild 是完全受控的持續整合服務，可編譯原始程式碼、執行測試，以及產生可供部署的軟體套件。透過 CodeBuild，您可以使用預先封裝的建置環境，或是建立使用自己建置工具的自訂建置環境。此示範案例會使用第二個選項。它由使用預先封裝的 Docker 映像檔的 CodeBuild 建置環境組成。

> **Important**
>
> 在您開始大型主機現代化專案之前，建議您先瞭解適用於大型主機的AWS Migration Acceleration Program (MAP)，或聯絡大型主機專家，瞭解AWS 大型主機應用程式現代化所需的步驟。

主題

- 必要條件
- 步驟 1：與 AWS 帳戶共享構建資產
- 步驟 2：創建 Amazon S3 存儲桶
- 步驟 3：建立建置規格檔案
- 步驟 4：上傳來源檔案

- 步驟 5：建立IAM策略
- 步驟 6：建立IAM角色
- 步驟 7：將IAM策略附加到IAM角色
- 步驟 8：建立 CodeBuild 專案
- 步驟 9：開始構建
- 步驟 10：下載輸出工件
- 清除資源

必要條件

開始此自學課程之前，請先完成下列先決條件。

- 下載BankDemo 範例應用程式並將其解壓縮至資料夾。源文件夾包含COBOL程序和字稿以及定義。它還包含一個JCL文件夾供參考，儘管您不需要構建JCL。該文件夾還包含構建所需的元文件。
- 在「 AWS 大型主機現代化」主控台中，選擇「工具」。在 [分析、開發和建置資產] 中，選擇 [與我的AWS帳戶共用資產]。

步驟 1：與 AWS 帳戶共享構建資產

在此步驟中，您必須確保與 AWS 帳戶共用組建資產，尤其是在使用資產的地區。

1. 在開啟大 AWS 型主機現代化主控台。https://console.aws.amazon.com/m2/
2. 在左側導覽中，選擇 [工具]。
3. 在 [分析、開發和建置資產] 中，選擇 [與我的 AWS 帳戶共用資產]。

> **Important**
>
> 您需要在您打算進行構建的每個 AWS 區域中執行此步驟一次。

步驟 2：創建 Amazon S3 存儲桶

在此步驟中，您會建立兩個 Amazon S3 儲存貯體。第一個是保存源代碼的輸入存儲桶，另一個是用於保存構建輸出的輸出存儲桶。如需詳細資訊，請參閱 Amazon S3 使用者指南中的建立、設定和使用 Amazon S3 儲存貯體。

1. 若要建立輸入儲存貯體，請登入 Amazon S3 主控台並選擇「建立儲存貯體」。
2. 在 [一般] 設定中，提供值區的名稱，並指定您 AWS 區域 要建立值區的位置。範例名稱為`codebuild-regionId-accountId-input-bucket`，其中`regionId`是值區 AWS 區域 的，而且`accountId`是您的 AWS 帳戶 ID。

   > **Note**
   >
   > 如果您在不同 AWS 區域 於美國東部 (維吉尼亞北部) 建立值區，請指定`LocationConstraint`參數。如需詳細資訊，請參閱 Amazon 簡單儲存服務API參考中的建立儲存貯體。

3. 保留所有其他設定並選擇 [建立值區]。
4. 重複步驟 1-3 以建立輸出值區。範例名稱為`codebuild-regionId-accountId-output-bucket`，其中`regionId`是值區 AWS 區域 的，而且`accountId`是您的 AWS 帳戶 ID。

   無論您為這些值區選擇什麼名稱，請務必在本教學課程中使用它們。

步驟 3：建立建置規格檔案

在此步驟中，您會建立組建規格檔案、. 此文件提供構建命令和相關設置，YAML格式，用 CodeBuild 於運行構建。如需詳細資訊，請參閱《使用指南》 CodeBuild中的〈建置規格參考AWS CodeBuild〉。

1. 在您解壓縮為先決條件`buildspec.yml`的目錄中建立一個名為的檔案。
2. 將下列內容新增至檔案並儲存。此檔案不需要變更。

```
version: 0.2
env:
  exported-variables:
    - CODEBUILD_BUILD_ID
    - CODEBUILD_BUILD_ARN
phases:
  install:
    runtime-versions:
      python: 3.7
  pre_build:
    commands:
      - echo Installing source dependencies...
      - ls -lR $CODEBUILD_SRC_DIR/source
```

```
  build:
    commands:
      - echo Build started on `date`
      - /start-build.sh -Dbasedir=$CODEBUILD_SRC_DIR/source -Dloaddir=
$CODEBUILD_SRC_DIR/target
  post_build:
    commands:
      - ls -lR $CODEBUILD_SRC_DIR/target
      - echo Build completed on `date`
artifacts:
  files:
    - $CODEBUILD_SRC_DIR/target/**
```

這裡CODEBUILD_BUILD_ID、、CODEBUILD_BUILD_ARN$CODEBUILD_SRC_DIR/source、和$CODEBUILD_SRC_DIR/target是可在其中使用的環境變數 CodeBuild。如需詳細資訊，請參閱建置環境中的環境變數。

此時，您的目錄應該是這樣的。

```
(root directory name)
    |-- build.xml
    |-- buildspec.yml
    |-- LICENSE.txt
    |-- source
         |... etc.
```

3. 將資料夾的內容壓縮到名為BankDemo.zip.. 在本教程中，您無法壓縮文件夾。而是將資料夾的內容壓縮到檔案中BankDemo.zip。

步驟 4：上傳來源檔案

在此步驟中，您將 BankDemo 範例應用程式的原始程式碼上傳到 Amazon S3 輸入儲存貯體。

1. 登入 Amazon S3 主控台，然後在左側導覽窗格中選擇「儲存貯體」。然後選擇您先前建立的輸入值區。
2. 在「物件」下選擇「上載」。
3. 在「檔案和資料夾」區段中，選擇「新增檔案」。
4. 瀏覽並選擇您的BankDemo.zip檔案。
5. 選擇上傳。

步驟 5：建立IAM策略

在此步驟中，您會建立兩個IAM原則。其中一個原則會授與 AWS 大型主機現代化的權限，以存取和使用包含 Micro Focus 建置工具的 Docker 映像檔。此政策不是為客戶量身定制的。其他政策授予 AWS 大型主機現代化與輸入和輸出值區以及產生的 Amazon CloudWatch 日誌互動的許可。 CodeBuild

若要瞭解如何建立IAM策略，請參閱IAM使用指南中的編輯IAM策略。

若要建立存取 Docker 映像檔的原則

1. 在IAM主控台中，複製下列原則文件並將其貼到原則編輯器中。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": "arn:aws:ecr:*:673918848628:repository/m2-enterprise-build-
tools"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::aws-m2-repo-*-<region>-prod"
        }
    ]
}
```

2. 提供原則的名稱，例如m2CodeBuildPolicy。

建立允許 AWS 大型主機現代化與值區和記錄互動的原則

1. 在IAM主控台中，複製下列原則文件並將其貼到原則編輯器中。請務必更新regionIdaccountId至和您的 AWS 帳戶. AWS 區域

```
{
      "Version": "2012-10-17",
      "Statement": [
          {
              "Action": [
                  "logs:CreateLogGroup",
                  "logs:CreateLogStream",
                  "logs:PutLogEvents"
              ],
              "Resource": [
                  "arn:aws:logs:regionId:accountId:log-group:/aws/codebuild/
codebuild-bankdemo-project",
                  "arn:aws:logs:regionId:accountId:log-group:/aws/codebuild/
codebuild-bankdemo-project:*"
              ],
              "Effect": "Allow"
          },
          {
              "Action": [
                  "s3:PutObject",
                  "s3:GetObject",
                  "s3:GetObjectVersion",
                  "s3:GetBucketAcl",
                  "s3:GetBucketLocation",
                  "s3:List*"
              ],
              "Resource": [
                  "arn:aws:s3:::codebuild-regionId-accountId-input-bucket",
                  "arn:aws:s3:::codebuild-regionId-accountId-input-bucket/*",
                  "arn:aws:s3:::codebuild-regionId-accountId-output-bucket",
                  "arn:aws:s3:::codebuild-regionId-accountId-output-bucket/*"
              ],
              "Effect": "Allow"
          }
      ]
  }
```

2. 提供原則的名稱，例如BankdemoCodeBuildRolePolicy。

步驟 6：建立IAM角色

在此步驟中，您會建立一個新IAM角色， CodeBuild 以便在您先前建立的IAM原則與此新IAM角色建立關聯之後，為您與 AWS 資源互動。

如需有關建立服務角色的資訊，請參閱《IAM使用指南》中的〈建立將權限委派給 AWS 服務的角色〉。

1. 登入IAM主控台，然後在左側導覽窗格中選擇 [角色]。
2. 選擇建立角色。
3. 在 [信任的實體類型] 下，選擇 [AWS服務
4. 在「其他AWS服務的使用案例」下 CodeBuild，選擇，然後CodeBuild再次選擇。
5. 選擇 Next (下一步)。
6. 在 Add permissions (新增許可) 頁面上，選擇 Next (下一步)。稍後您可以將原則指派給角色。
7. 在「角色詳細資料」下，提供角色的名稱，例如，BankdemoCodeBuildServiceRole。
8. 在 [選取信任的實體] 底下，確認原則文件如下所示：

```
{
          "Version": "2012-10-17",
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": "codebuild.amazonaws.com"
              },
              "Action": "sts:AssumeRole"
            }
          ]
        }
```

9. 選擇建立角色。

步驟 7：將IAM策略附加到IAM角色

在此步驟中，您會將先前建立的兩個IAM原則附加至BankdemoCodeBuildServiceRoleIAM角色。

1. 登入IAM主控台，然後在左側導覽窗格中選擇 [角色]。
2. 在「角色」中，選擇您先前建立的角色，例如，BankdemoCodeBuildServiceRole。
3. 在 [權限] 原則中，選擇 [新增權限]，然後選擇 [附加原則
4. 在 [其他權限原則] 中，選擇您先前建立的原則，例如m2CodeBuildPolicy和BankdemoCodeBuildRolePolicy。
5. 選擇連接政策。

步驟 8：建立 CodeBuild 專案

在此步驟中，您將建立 CodeBuild 專案。

1. 登入 CodeBuild 主控台並選擇 [建立組建專案]。
2. 在 [專案組態] 區段中，提供專案的名稱，例如codebuild-bankdemo-project。
3. 在「來源」區段中，針對來源供應商選擇 Amazon S3，然後選擇您先前建立的輸入儲存貯體，例如codebuild-regionId-accountId-input-bucket。
4. 在 S3 物件金鑰或 S3 資料夾欄位中，輸入您上傳到 S3 儲存貯體的 zip 檔案名稱。在此情況下，檔案名稱為bankdemo.zip。
5. 在「環境」區段中，選擇「自訂影像」。
6. 在「環境類型」欄位中，選擇「Linux」。
7. 在 [影像登錄] 下，選擇 [其他登錄]
8. 在「外部登錄 URL」欄位中，
   - 對於微對焦 v9：輸入673918848628.dkr.ecr.us-west-1.amazonaws.com/m2-enterprise-build-tools:9.0.7.R1。如果您使用 Micro Focus v9 的其他 AWS 區域，您也可以指定 673918848628.dkr.ecr.<m2-region>.amazonaws.com/m2-enterprise-build-tools:9.0.7.R1<m2-region>提供 AWS 大型主機現代化服務的 AWS 區域在哪裡 (例如)。eu-west-3
   - 對於微型對焦 v8：輸入 673918848628.dkr.ecr.us-west-2.amazonaws.com/m2-enterprise-build-tools:8.0.9.R1
   - 對於微型對焦 v7：輸入 673918848628.dkr.ecr.us-west-2.amazonaws.com/m2-enterprise-build-tools:7.0.R10
9. 在 [服務角色] 底下，選擇 [現有服務角色ARN]，然後在 [角色] 欄位中選擇您先前建立的服務角色，例如BankdemoCodeBuildServiceRole。

10. 在 [建置規格] 區段中，選擇 [使用組建規格檔案]。
11. 在「成品」區段的「類型」下，選擇 Amazon S3，然後選擇輸出儲存貯體，例如`codebuild-regionId-accountId-output-bucket`。
12. 在「名稱」欄位中，輸入值區中要包含組建輸出成品的資料夾名稱，例如`bankdemo-output.zip`。
13. 在「成品包裝」下，選擇「Zip」。
14. 選擇 Create build project (建立建置專案)。

步驟 9：開始構建

在此步驟中，您將啟動組建。

1. 登入 CodeBuild 主控台。
2. 在左側導覽窗格中，選擇 [建置專案]。
3. 選擇您先前建立的建置專案，例如`codebuild-bankdemo-project`。
4. 選擇 Start build (開始組建)。

此命令啟動構建。組建會以非同步方式執行。命令的輸出是包JSON含屬性 id 的輸出。此屬性 idis 對剛剛啟動之組 CodeBuild 建 ID 的參考。您可以在 CodeBuild 主控台中檢視組建的狀態。您還可以在控制台中查看有關構建執行的詳細日誌。如需詳細資訊，請參閱AWS CodeBuild 使用者指南中的檢視詳細組建資訊。

當目前階段為時COMPLETED，表示您的建置已成功完成，且編譯的成品已在 Amazon S3 上準備就緒。

步驟 10：下載輸出工件

在此步驟中，您會從 Amazon S3 下載輸出成品。Micro Focus 建置工具可以建立數種不同的可執行檔類型。在此自學課程中，它會產生共用物件。

1. 登入 Amazon S3 主控台。
2. 在「值區」角色 =「粗體」> 區段中，選擇輸出值區的名稱，例如，。`codebuild-regionId-accountId-output-bucket`
3. 選擇 [下載角色] = [粗體] >。
4. 解壓縮所下載的 檔案。導覽至目標資料夾以查看組建加工品。其中包括 `.so` Linux 共享對象。

清除資源

如果您不再需要為此教學課程建立的資源，請刪除這些資源以免產生額外費用。若要這樣做，請完成下列步驟：

- 刪除您為本教學課程建立的 S3 儲存貯體。如需詳細資訊，請參閱 Amazon 簡單儲存服務使用者指南中的刪除儲存貯體。
- 刪除您為此教學課程建立的IAM策略。如需詳細資訊，請參閱《使用指南》中的IAM〈刪除IAM策略〉。
- 刪除您為此教學課程建立的IAM角色。如需詳細資訊，請參閱《IAM使用指南》中的刪除角色或執行個體設定檔。
- 刪除您為此自學課程建立的 CodeBuild 專案。如需詳細資訊，請參閱《使用指南》 CodeBuild中的〈刪除組建專案AWS CodeBuild〉。

教學課程：設定 CI/CD 管道，以便與微焦點企業開發人員搭配使用

本教學課程說明如何在 Micro Focus 企業開發人員中匯入、編輯、編譯和執行 BankDemo 範例應用程式，然後認可變更以觸發 CI/CD 管線。

內容

- 必要條件
- 建立 CI/CD 管線基本基礎架構
- 建立 AWS CodeCommit 儲存庫和 CI/CD 管道
  - 範例YAML觸發程式檔案組態 git.yml
- 企業開發人員 AppStream 2.0 建立
- 企業開發人員設置和測試
  - 在企業開發人員中克隆 BankDemo CodeCommit 存儲庫
  - 建立 BankDemo 大型主機專案並建置應用COBOL程式
  - 創建本地 BankDemo CICS和批處理環境進行測試
  - 從企業開發人員啟動BANKDEMO伺服器
  - 啟動倫巴 3270 終端機
  - 執行交 BankDemo易
  - 從企業開發人員停止BANKDEMO服務器
- 練習一：在BANKDEMO申請中加強貸款計算

  - 將貸款分析規則新增至企業開發人員程式碼
  - 步驟 1：執行貸款計算的代碼分析
  - 步驟 2：修改CICSBMS地圖和COBOL程序並進行測試
  - 步驟 3：在COBOL程序中添加總金額計算
  - 步驟 4：提交更改並運行 CI/CD 管道
- 練習 2：在 BankDemo申請中提取貸款計算
  - 第 1 步：重構貸款計算常式到一個COBOL部分
  - 步驟 2：將貸款計算程序提取到獨立COBOL程序
  - 步驟 3：提交變更並執行 CI/CD 管線
- 清除資源

必要條件

下載下列檔案。

- basic-infra.yaml
  - 從歐洲（法蘭克福）地區下載。
  - 從美國東部 (維吉尼亞北部) 區域下載。
- pipeline.yaml
  - 從歐洲（法蘭克福）地區下載。
  - 從美國東部 (維吉尼亞北部) 區域下載。
- m2-code-sync-function.zip
  - 從歐洲（法蘭克福）地區下載。
  - 從美國東部 (維吉尼亞北部) 區域下載。
- config_git.yml
  - 從歐洲（法蘭克福）地區下載。
  - 從美國東部 (維吉尼亞北部) 區域下載。
- BANKDEMO-source.zip
  - 從歐洲（法蘭克福）地區下載。
  - 從美國東部 (維吉尼亞北部) 區域下載。
- BANKDEMO-exercise.zip
  - 從歐洲（法蘭克福）地區下載。

- 從美國東部 (維吉尼亞北部) 區域下載。

每個文件的目的如下：

`basic-infra.yaml`

此 AWS CloudFormation 範本會建立 CI/CD 管道所需的基本基礎設施：VPC、Amazon S3 儲存貯體等。

`pipeline.yaml`

Lambda 函數會使用此 AWS CloudFormation 範本來啟動管線堆疊。請確定此範本位於可公開存取的 Amazon S3 儲存貯體中。將此值區的連結新增為`basic-infra.yaml`範本中`PipelineTemplateURL`參數的預設值。

`m2-code-sync-function.zip`

此 Lambda 函數會建立 CodeCommit 儲存庫 (以中為基礎的目錄結構)`config_git.yaml`，並使用啟動管線堆疊`pipeline.yaml`。請確定所有支援 AWS 大型主機現代化的可公開存取 Amazon S3 儲存貯體中都 AWS 區域 可使用此 zip 檔案。我們建議您將檔案儲存在值區中， AWS 區域 並將其複製到所有儲存貯體中 AWS 區域。使用具有可識別特定字尾的值區命名慣例 AWS 區域 (例如，`m2-cicd-deployment-source-eu-west-1`)，並將字首新增`m2-cicd-deployment-source`為參數的預設值，`DeploymentSourceBucket`並在資源範`basic-infra.yaml`本中參照該值區`!Sub {DeploymentSourceBucket}-${AWS::Region}`時，使用 AWS CloudFormation 替代函數形成完整值區`SourceSyncLambdaFunction`。

`config_git.yml`

CodeCommit 目錄結構定義。如需詳細資訊，請參閱範例YAML觸發程式檔案組態 git.yml。

`BANKDEMO-source.zip`.

BankDemo 源代碼和從 CodeCommit存儲庫創建的配置文件。

`BANKDEMO-exercise.zip`.

BankDemo 從 CodeCommit 儲存庫建立的自學課程練習的來源。

建立 CI/CD 管線基本基礎架構

使用 AWS CloudFormation 範`basic-infra.yaml`本透過主控台建立 CI/CD 管線基本基礎結構堆疊。 AWS CloudFormation 此堆疊會建立 Amazon S3 儲存貯體，您可以在其中上傳應用程式程式碼

和資料，以及建立其他必要資源 (例如 AWS CodeCommit 存放庫和 AWS CodePipeline 管道) 的支援 AWS Lambda 功能。

> **Note**
>
> 若要啟動此堆疊，您需要具有管理權限IAM、Amazon S3、Lambda AWS CloudFormation 和許可才能使用 AWS KMS。

1. 登入 AWS Management Console 並在 https://console.aws.amazon.com/cloudFormation 中開啟 AWS CloudFormation 主控台。
2. 使用以下選項之一建立新的堆疊：
   - 選擇 Create Stack (建立堆疊)。如果您目前有堆疊正在執行，這是「唯一的」選項。
   - 在「堆疊」頁面上，選擇「建立堆疊」。只有當無堆疊執行時才看得到此選項。
3. 在「指定樣板」頁面上：
   - 在「準備範本」中，選擇「範本已就緒」。
   - 在指定範本中，選擇 Amazon S3 URL 做為範本來源，然後URLs根據您的 AWS 區域.
     - `https://m2-us-east-1.s3.us-east-1.amazonaws.com/cicd/mf/basic-infra.yaml`
     - `https://m2-eu-central-1.s3.eu-central-1.amazonaws.com/cicd/mf/basic-infra.yaml`
   - 若要接受設定，請選擇 [下一步]。

   [建立堆疊] 頁面隨即開啟。

Specify stack details

Stack name

Stack name

m2-cicd-pipeline-foundation

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

Networking Configuration

Do you want to use an existing VPC in your account?

If you select 'Yes', then you must provide the VPC ID and the Subnet IDs.

No (Create one)

Which VPC ID should be used?

If you selected 'Yes' for UseExistingVPC, this parameter is required. Otherwise, this value will be ignored.

Which private subnet ID should be used?

If you selected 'Yes' for UseExistingVPC, this parameter is required. Otherwise, this value will be ignored.

Which private subnet ID in a different AZ should be used for HA?

If you selected 'Yes' for UseExistingVPC, this parameter is required. Otherwise, this value will be ignored.

Enter the CIDR block that should be used for the new VPC

If you selected 'No (Create one)' for UseExistingVPC, this parameter is required. Otherwise, this value will be ignored.

11.11.0.0/16

CIDR bits for creating subnets. Choose 5 for /27, 6 for /26, 7 for /25, 8 for /24 range

If you selected 'No (Create one)' for UseExistingVPC, this parameter is required. Otherwise, this value will be ignored.

5

Deployment Configuration

Name of the S3 bucket which contains the source files for this stack deployment

Don't change unless you know what you are doing.

m2

Name of the source package file for the infrastructure Lambda function

Don't change unless you know what you are doing.

cicd/mf/m2-code-sync-function.zip

Full URL of the pipeline CloudFormation template file

Don't change unless you know what you are doing.

https://m2-us-east-1.s3.us-east-1.amazonaws.com/cicd/mf/pipeline.yaml

What name prefix to use for the new S3 buckets?

A name prefix for the S3 buckets that will be created by this stack.

mf

Cancel Previous Next

進行下列變更：

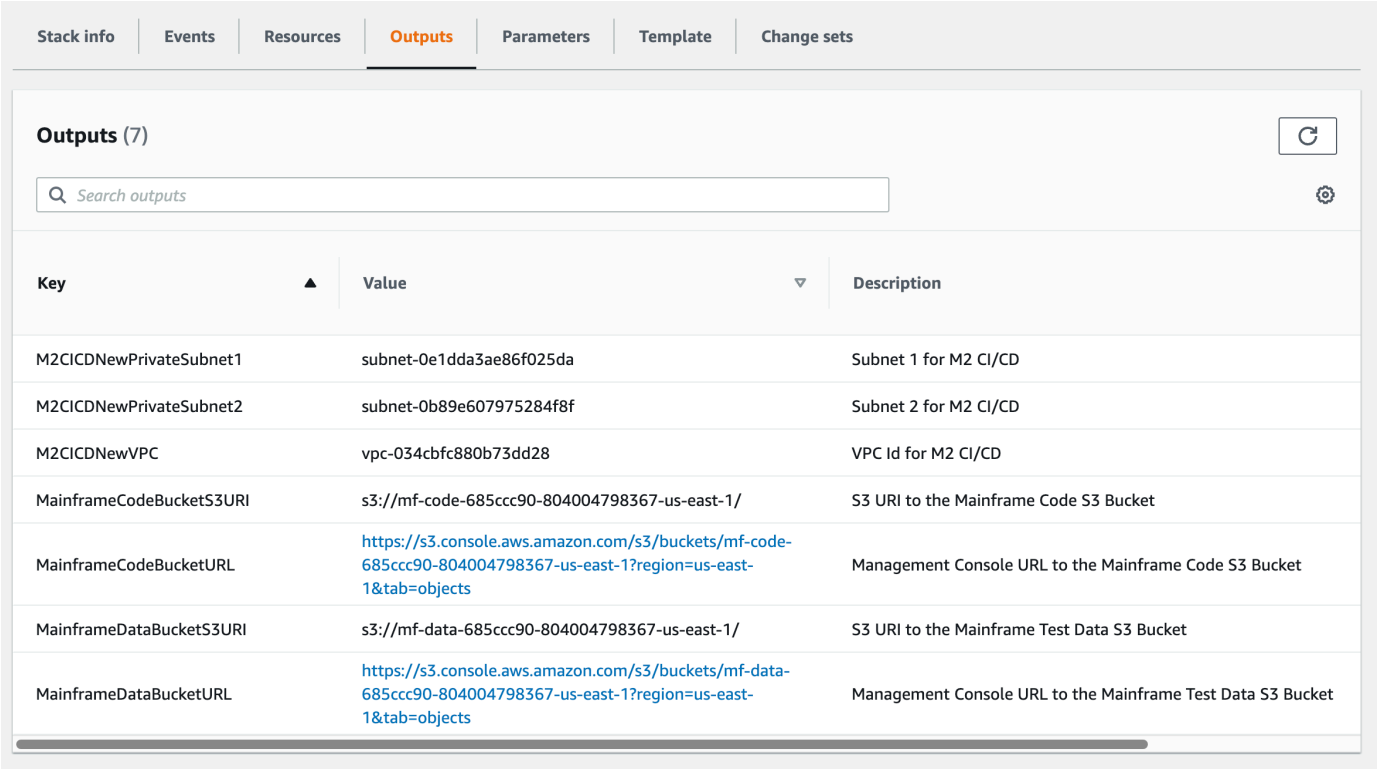
- 為網路組態的堆疊名稱和參數提供適當的值。
- 部署組態中的大多數參數都會適當地預先填入，因此您不需要修改它們。根據您的管道範本 AWS 區域，將管道 AWS CloudFormation 範本變更為下列 Amazon S3 之一URLs。
  - `https://m2-us-east-1.s3.amazonaws.com/cicd/mf/pipeline.yaml`
  - `https://m2-eu-central-1.s3.eu-central-1.amazonaws.com/cicd/mf/pipeline.yaml`
- 選擇 Next (下一步)。

> **Note**
>
> 除非您已自行修改 AWS CloudFormation 範本，否則請勿變更預設參數值。

4. 在設定堆疊選項中，選擇下一步。
5. 在 [權能] 中，選擇 [我確認 AWS CloudFormation 可能會建立IAM資源，以允許代表您建立IAM角色] 的權限。 AWS CloudFormation 選擇建立堆疊。

> **Note**
>
> 佈建此堆疊可能需要 3 到 5 分鐘。

6. 成功建立堆疊之後，瀏覽至新佈建之堆疊的「輸出」區段。您會在那裡找到需要上傳大型主機程式碼和相依檔案的 Amazon S3 儲存貯體。



建立 AWS CodeCommit 儲存庫和 CI/CD 管道

在此步驟中，您可以呼叫 Lambda 函數 AWS CloudFormation 來建立管線堆疊，以建立 CodeCommit 存放庫並佈建 CI/CD 管線堆疊。

1. 將BankDemo 範例應用程式下載到您的本機電腦。
2. `bankdemo.zip`從您的本機電腦上傳到在中建立的 Amazon S3 儲存貯體建立 CI/CD 管線基本基礎架構。
3. 下載 `config_git.yml`。
4. 視需要修改，`config_git.yml`如下所示：

   - 添加您自己的目標存儲庫名稱，目標分支和提交消息。

   ```
   repository-config:
     target-repository: bankdemo-repo
     target-branch: main
     commit-message: Initial commit for bankdemo-repo main branch
   ```

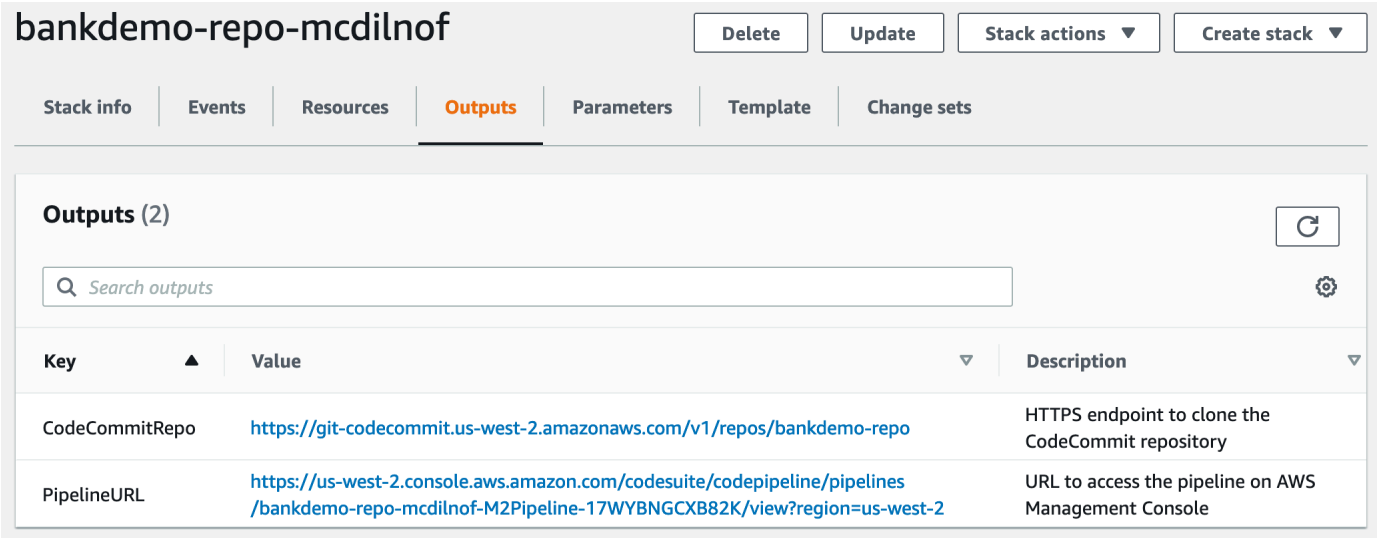
   - 新增您要接收通知的電子郵件地址。

```
pipeline-config:
  # Send pipeline failure notifications to these email addresses
  alert-notifications:
    - myname@mycompany.com
  # Send notifications for manual approval before production deployment to these
 email addresses
  approval-notifications:
    - myname@mycompany.com
```

5. 將包含 CodeCommit 存放庫資料夾結構定義的`config_git.yml`檔案上傳到在中建立的 Amazon S3 儲存貯體建立 CI/CD 管線基本基礎架構。這將叫用將自動佈建存放庫和管道的 Lambda 函數。

   這將使用在`config_git.yml`文件中`target-repository`定義的名稱創建一個 CodeCommit 存儲庫；例如，`bankdemo-repo`。

   Lambda 函數也會透過建立 CI/CD 管線堆疊。 AWS CloudFormation AWS CloudFormation 堆棧將具有相同的前綴提供的`target-repository`名稱，後跟隨機字符串（例如`bankdemo-repo-`*`01234567`*。您可以在「管理主控台」URL 中找到 CodeCommit 存放庫URL和存取建立的 AWS 管道。



6. 如果 CodeCommit 存放庫建立完成，將立即觸發 CI/CD 管線以執行完整的 CI/CD。
7. 推送文件後，它將自動觸發管道，該管道將構建，在測試中部署，運行一些測試並等待手動批准，然後將其部署到生產環境中。

範例YAML觸發程式檔案組態 git.yml

```
repository-config:
```

```
target-repository: bankdemo-repo
target-branch: main
commit-message: Initial commit for bankdemo-repo main branch
directory-structure:
  - '/':
      files:
      - build.xml
      - '*.yaml'
      - '*.yml'
      - '*.xml'
      - 'LICENSE.txt'
      readme: |
        # Root Folder
        - 'build.xml' : Build configuration for the application
  - tests:
      files:
      - '*.py'
      readme: |
        # Test Folder
        - '*.py' : Test scripts
  - config:
      files:
      - 'BANKDEMO.csd'
      - 'BANKDEMO.json'
      - 'BANKDEMO_ED.json'
      - 'dfhdrdat'
      - 'ESPGSQLXA.dll'
      - 'ESPGSQLXA64.so'
      - 'ESPGSQLXA64_S.so'
      - 'EXTFH.cfg'
      - 'm2-2021-04-28.normal.json'
      - 'MFDBFH.cfg'
      - 'application-definition-template-config.json'
      readme: |
        # Config Folder
        This folder contains the application configuration files.
        - 'BANKDEMO.csd'     : CICS Resource definitions export file
        - 'BANKDEMO.json'    : Enterprise Server configuration
        - 'BANKDEMO_ED.json' : Enterprise Server configuration for ED
        - 'dfhdrdat'         : CICS resource definition file
        - 'ESPGSQLXA.dll'    : XA switch module Windows
        - 'ESPGSQLXA64.so'   : XA switch module Linux
        - 'ESPGSQLXA64_S.so' : XA switch module Linux
        - 'EXTFH.cfg'        : Micro Focus File Handler configuration
```

```
          - 'm2-2021-04-28.normal.json' : M2 request document
          - 'MFDBFH.cfg'       : Micro Focus Database File Handler
          - 'application-definition-template-config.json' : Application definition for
 M2
    - source:
        subdirs:
        - .settings:
            files:
              - '.bms.mfdirset'
              - '.cbl.mfdirset'
        - copybook:
            files:
              - '*.cpy'
              - '*.inc'
            readme: |
              # Copy folder
              This folder contains the source for COBOL copy books, PLI includes, ...
              - .cpy COBOL copybooks
              - .inc PLI includes
#        - ctlcards:
#            files:
#              - '*.ctl'
#              - 'KBNKSRT1.txt'
#            readme: |
#              # Control Card folder
#              This folder contains the source for Batch Control Cards
#              - .ctl Control Cards
        - ims:
            files:
              - '*.dbd'
              - '*.psb'
            readme: |
              # ims folder
              This folder contains the IMS DB source files with the extensions
              - .dbd for IMS DBD source
              - .psb for IMS PSB source
        - jcl:
            files:
              - '*.jcl'
              - '*.ctl'
              - 'KBNKSRT1.txt'
              - '*.prc'
            readme: |
              # jcl folder
```

```
              This folder contains the JCL source files with the extensions
              - .jcl
#        - proclib:
#            files:
#              - '*.prc'
#            readme: |
#              # proclib folder
#              This folder contains the JCL procedures referenced via PROCLIB
 statements in the JCL with extensions
#              - .prc
        - rdbms:
            files:
              - '*.sql'
            readme: |
              # rdbms folder
              This folder contains any DB2 related source files with extensions
              - .sql for any kind of SQL source
        - screens:
            files:
            - '*.bms'
            - '*.mfs'
            readme: |
              # screens folder
              This folder contains the screens source files with the extensions
              - .bms for CICS BMS screens
              - .mfs for IMS MFS screens
            subdirs:
            - .settings:
                files:
                    - '*.bms.mfdirset'
        - cobol:
            files:
              - '*.cbl'
              - '*.pli'
            readme: |
              # source folder
              This folder contains the program source files with the extensions
              - .cbl for COBOL source
              - .pli for PLI source
            subdirs:
            - .settings:
                files:
                   - '*.cbl.mfdirset'
    - tests:
```

```
        files:
        - 'test_script.py'
        readme: |
          # tests Folder
          This folder contains the application test scripts
pipeline-config:
  alert-notifications:
    - myname@mycompany.com
  approval-notifications:
    - myname@mycompany.com
```

企業開發人員 AppStream 2.0 建立

若要在 AppStream 2.0 上設定微焦企業開發人員，請參閱教學課程：在 AppStream 2.0 上設定微焦點企業開發人員。

若要將 CodeCommit 存放庫連線至企業開發人員，請使用`target-repository`中指定的名稱範例 YAML觸發程式檔案組態 git.yml。

企業開發人員設置和測試

主題

- 在企業開發人員中克隆 BankDemo CodeCommit 存儲庫
- 建立 BankDemo 大型主機專案並建置應用COBOL程式
- 創建本地 BankDemo CICS和批處理環境進行測試
- 從企業開發人員啟動BANKDEMO伺服器
- 啟動倫巴 3270 終端機
- 執行交 BankDemo易
- 從企業開發人員停止BANKDEMO服務器

Connect 到您在中建立的企業開發人員 AppStream 2.0 執行個體企業開發人員 AppStream 2.0 建立。

1. 從 Windows 開始啟動企業開發人員。選擇微焦點企業開發人員，然後選擇 Eclipse 的企業開發人員。如果您是第一次開始，可能需要一些時間。
2. 在 Eclipse 啟動器的「工作區：輸入」，`C:\Users\<username>\workspace`然後選擇「啟動」。

> **Note**
>
> 請務必在重新連線至 AppStream 2.0 執行個體後選擇相同的位置。工作區選取不是永久性的。

3. 在歡迎中，選擇開啟COBOL透視圖。這只會在新工作區的第一次顯示。

在企業開發人員中克隆 BankDemo CodeCommit 存儲庫

1. 選擇「視窗」/「透視」/「開啟透視/其他... /Git。
2. 選擇複製 Git 儲存庫。
3. 在複製 Git 儲存庫中，輸入下列資訊：
   - 在位置中URI，輸入 CodeCommit存放庫HTTPSURL的。

     > **Note**
     >
     > 在「 AWS 管理主控台」中複製 CodeCommit 儲存區域的複製，然後將其貼到此處。URL HTTPS將URI會分割成主機和儲存庫路徑。

   - 驗證使用者和密碼中的使用者 CodeCommit 儲存庫認證，然後選擇儲存在安全存放區中。
4. 在「分支選擇」中，選擇「主分支」，然後選擇「下一步
5. 在本機目的地的目錄中，輸入`C:\Users\<username>\workspace`並選擇完成。

   當「Git 存放庫」檢視中顯示時，`BANKDEMO [main]`就會完成複製程序。

建立 BankDemo 大型主機專案並建置應用COBOL程式

1. 變更為「COBOL透視」。
2. 在專案中，停用「自動建置」。
3. 在檔案中，選擇新增，然後選擇大型主機COBOL專案。
4. 在新大型主機COBOL專案中，輸入下列資訊：
   - 在專案名稱中，輸入`BankDemo`。
   - 選擇微型對焦範本 [64 位元]。
   - 選擇 Finish (完成)。

5. 在COBOL檔案總管中，展開新 BankDemo 專案。

   > **Note**
   >
   > [BANKDEMO main]方括號表示該項目與本地 BankDemo CodeCommit 存儲庫連接。

6. 如果樹狀檢視未顯示 [COBOL程式集]、[撰寫本]、[BMS來源] 和 [JCL檔案] 的項目，請從 BankDemo 專案快顯功能表中選擇 [重新整理]。
7. 從 BankDemo 內容功能表中，選擇「內容」/「微型對焦」/「專案設定」/COBOL：
   - 選擇「字元集」-ASCII。
   - 選擇套用，然後選擇關閉。
8. 如果BMS和COBOL來源的建置並未立即啟動，請在 [專案] 功能表中簽入 [自動建置] 選項已啟用。

   Build 輸出將顯示在 Console 視圖中，並且應該在幾分鐘後完成，其中包含消息BUILD SUCCESSFUL和Build finished with no errors。

   應用 BankDemo 程序現在應該被編譯並準備好進行本地執行。

創建本地 BankDemo CICS和批處理環境進行測試

1. 在COBOL檔案總管中，展開BANKDEMO / config。
2. 在編輯器中，開啟BANKDEMO_ED.json。
3. 查找字符串ED_Home=並更改路徑以指向企業開發人員項目，如下所示：D:\\<username>\\workspace\\BANKDEMO。請注意在路徑定義中使用雙斜線 (\\)。
4. 儲存並關閉檔案。
5. 選擇伺服器總管。
6. 從「預設」內容功能表中，選擇「開啟管理頁面」。「Micro Focus 企業伺服器管理」頁面會在預設瀏覽器中開啟。
7. 僅對於 AppStream 2.0 個工作階段，請進行下列變更，以便保留本機企業伺服器區域以進行本機測試：
   - 在目錄伺服器/預設值中，選擇 PROPERTIES/組態。
   - 將儲存庫位置取代為D:\<username>\My Files\Home Folder\MFDS。

> **Note**
>
> 每次連接 AppStream 2.0 執行個體後，您必須完成步驟 5-8。

8. 在目錄伺服器/預設值中，選擇匯入，然後完成下列步驟：

   - 在步驟 1：匯入類型中，選擇JSON並選擇下一步。
   - 在「步驟 2：上傳」中，按一下以上傳藍色方塊中的檔案。
   - 在選擇要上傳的檔案中，輸入：
     - 檔案名稱:`D:\<username>\workspace\BANKDEMO\config\BANKDEMO_ED.json`.
     - 選擇 Open (開啟)。
   - 選擇 Next (下一步)。
   - 在「步驟 3：區域」中清除「清除端點的連接埠」。
   - 選擇 Next (下一步)。
   - 在「步驟 4：匯入」中，選擇「匯入」。
   - 選擇 Finish (完成)。

   清單現在會顯示新的伺服器名稱BANKDEMO。

從企業開發人員啟動BANKDEMO伺服器

1. 選擇企業開發人員。
2. 在 [伺服器總管] 中，選擇 [預設]，然後從內容功能表選擇 [重新整

   伺服器清單現在也應該會顯示BANKDEMO。
3. 選擇BANKDEMO。
4. 從內容功能表中選擇「與專案關聯」，然後選擇BANKDEMO。
5. 從內容功能表中選擇 [開始]。

   控制台視圖應顯示服務器啟動的日誌。

   如果顯示訊息`BANKDEMO CASSI5030I PLTPI Phase 2 List(PI) Processing Completed`，表示伺服器已準備好測試CICSBANKDEMO應用程式。

啟動倫巴 3270 終端機

1. 從 Windows 開始，啟動微焦點倫巴 + 桌上型電腦/倫巴 + 桌上型電腦。
2. 在歡迎中，選擇 CREATENEWSESSION/大型主機顯示器。
3. 在「大型主機顯示器」中，選擇「連線/設定」。
4. 在 [工作階段組態] 中，選擇 [連線/TN3270]
5. 在 [主機名稱/位址] 中，選擇 [插入] 並輸入 IP 位址`127.0.0.1`。
6. 在 Telnet 連接埠中，輸入連接埠`6000`。
7. 選擇套用。
8. 選擇連線。

   CICS歡迎畫面會顯示畫面，其中包含第 1 列訊息：`This is the Micro Focus MFE CICS region BANKDEMO`。
9. 按 CTRL + 轉移 +Z 清除屏幕。

執行交 BankDemo易

1. 在空白螢幕中，輸入BANK。
2. 在屏幕 BANK10 中，在用戶 ID 的輸入字段中... ：，輸入guest並按下輸入。
3. 在屏幕 BANK20 中，在計算貸款成本之前的輸入字段中，輸入/（正斜杠），然後按 Enter 鍵。
4. 在畫面 BANK70 中：
   - 在您想借入的金額... ：，輸入`10000`。
   - 在以... 的利率計算 ：，輸入`5.0`。
   - 在多少個月... ：，輸入`10`。
   - 按 Enter。

   應該會顯示下列結果：

```
Resulting monthly payment.............:   $1023.06
```

這樣就完成了企業開發人員的BANKDEMO應用程式設定

從企業開發人員停止BANKDEMO服務器

1. 在 [伺服器總管] 中，選擇 [預設]，然後從內容功能表選擇 [重新整
2. 選擇BANKDEMO。
3. 從內容功能表中選擇「停止」。

   控制台視圖應顯示服務器停止的日誌。

   如果顯示訊息`Server: BANKDEMO stopped successfully`，表示伺服器已成功關閉。

練習一：在BANKDEMO申請中加強貸款計算

主題

- 將貸款分析規則新增至企業開發人員程式碼
- 步驟 1：執行貸款計算的代碼分析
- 步驟 2：修改CICSBMS地圖和COBOL程序並進行測試
- 步驟 3：在COBOL程序中添加總金額計算
- 步驟 4：提交更改並運行 CI/CD 管道

在這個案例中，您會逐步執行對程式碼進行範例變更、部署和測試程式碼的程序。

貸款部門希望在「貸款計算」屏幕 BANK7 0 上創建一個新字段以顯示「總貸款金額」。這需要更改屏BMS幕 MBANK7 0。 CBL，添加一個新的字段和相應的屏幕處理程序 SBANK7 0P。 CBL與相關的字筆。此外，貸款計算程序在 BBANK7 0P。 CBL需要使用額外的公式進行擴展。

若要完成此練習，請確定您已完成下列先決條件。

- 下載 BANKDEMO-exercise.zip 到`D:\PhotonUser\My Files\Home Folder`.
- 將 zip 檔案解壓縮至`D:\PhotonUser\My Files\Home Folder\BANKDEMO-exercise`。
- 建立資料夾`D:\PhotonUser\My Files\Home Folder\AnalysisRules`。
- 將規則檔案`Loan+Calculation+Update.General-1.xml`從`BANKDEMO-exercise`資料夾複製到`D:\PhotonUser\My Files\Home Folder\AnalysisRules`。

> **Note**
>
> * 中的程式碼變更。 CBL和 *. CPY在本練習中，在第 1 至 6 欄中以 EXER 01 標記。

將貸款分析規則新增至企業開發人員程式碼

在 Micro Focus 企業分析器中定義的分析規則可以從企業分析器導出，並導入到企業開發人員在企業開發人員項目中的源運行相同的分析規則。

1. 打開 `Window/Preferences/Micro Focus/COBOL/Code Analysis/Rules`.
2. 選擇編輯... 然後輸入`D:\PhotonUser\My Files\Home Folder\AnalysisRules`包含 rules 檔案的資料夾名稱`Loan+Calculation+Update.General-1.xml`。
3. 選擇 Finish (完成)。
4. 選擇套用，然後選擇關閉。
5. 從BANKDEMO專案關聯式功能表中，選擇 [程式碼分析]。

   您應該會看到 「貸款計算更新」 的項目。

步驟 1：執行貸款計算的代碼分析

有了新的分析規則，我們想要識別那裡匹配搜索模式的COBOL程序和代碼行*PAYMENT*，以*LOAN*及*RATE*在表達式，語句和變量。這將有助於瀏覽代碼並識別所需的代碼更改。

1. 從BANKDEMO專案右鍵功能表中，選擇「程式碼分析/貸款計算更新」。

   這將運行搜索規則，並在名為代碼分析的新選項卡中列出結果。當右下角的綠色進度列消失時，即完成分析執行。

   [程式碼分析] 索引標籤應該會顯示一個展開的清單 `BBANK20P.CBL` `BBANK70P.CBLSBANK70P.CBL`，其中每個清單都會列出符合搜尋模式的陳述式、運算式和變數。

   查看結果，只`BBANK20P.CBL`有移動的文字與搜索模式匹配。所以這個程序可以被忽略。
2. 在選項卡菜單欄中選擇 -圖標全部折疊。
3. 通過雙擊以任意順序展開`SBANK70P.CBL`並選擇任何行，以查看如何打開源代碼並突出顯示在源代碼中選擇的行。您還將識別出所有標識的源行都被標記。

步驟 2：修改CICSBMS地圖和COBOL程序並進行測試

首先，我們將更改BMS地圖MBANK70.BMS和屏幕處理程序SBANK70P.CBL和字帖CBANKDAT.CPY以顯示新字段。為了避免在本練習中不必要的編碼，修改過的來源模組可在D:\PhotonUser\My Files\Home Folder\BANKDEMO-exercise\Exercise01資料夾中使用。通常，開發人員會使用代碼分析結果來導航和修改源。如果您有時間並且想要進行手動更改，請使用 * 手動更改中提供的信息進行操作MBANK7。 BMS和 SBANK7 0 P. CBL(選擇性) *.

若要快速變更，請複製下列檔案：

1. ..\BANKDEMO-exercise\Exercis01\screens\MBANK70.BMS 至 D:\PhotonUser\workspace\bankdemo\source\screens。
2. .\BANKDEMO-exercise\Exercis01\cobol\SBANK70P.CBL 至 D:\PhotonUser\workspace\bankdemo\source\cobol。
3. ..\BANKDEMO-exercise\Exercis01\copybook\CBANKDAT.CPY 至 D:\PhotonUser\workspace\bankdemo\source\copybook。
4. 若要確保編譯受變更影響的所有程式，請選擇 [專案/清除...]。 /清理所有項目。

若要手動變更MBANK70.BMS和SBANK70P.CBL，請完成以下步驟：

- 對於在BMSMBANK70.BMS源代碼中手動更改，請在PAYMENT字段後添加：
  - TXT09 屬性與 TXT 08 相同，INITIAL價值為「總貸款額」
  - TOTAL具有相同的屬性 PAYMENT

測試變更

若要測試變更，請重複下列各節中的步驟：

1. 從企業開發人員啟動BANKDEMO伺服器
2. 啟動倫巴 3270 終端機
3. 執行交 BankDemo易

   此外，您現在還應該看到文本Total Loan Amount.....:。
4. 從企業開發人員停止BANKDEMO服務器

步驟 3：在COBOL程序中添加總金額計算

在第二步中，我們將更改BBANK70P.CBL並添加總貸款金額的計算。具有所需更改的準備源可在D:\PhotonUser\My Files\Home Folder\BANKDEMO-exercise\Exercise01文件夾中找到。如果您有時間並希望進行手動更改，請使用 BBANK7 0P 中提供的信息進行手動更改。 CBL(選擇性) *.

為了快速更改，請複製以下文件：

- ..\BANKDEMO-exercise\Exercis01\source\cobol\BBANK70P.CBL 至 D:\PhotonUser\workspace\bankdemo\source\cobol。

若要手動變更為BBANK70P.CBL，請完成以下步驟：

- 使用程式碼分析結果來識別所需的變更。

測試變更

若要測試變更，請重複下列各節中的步驟：

1. 從企業開發人員啟動BANKDEMO伺服器
2. 啟動倫巴 3270 終端機
3. 執行交 BankDemo易

   此外，您現在還應該看到文本Total Loan Amount.....: $10230.60。
4. 從企業開發人員停止BANKDEMO服務器

步驟 4：提交更改並運行 CI/CD 管道

將變更提交至中央 CodeCommit 儲存庫，並觸發 CI/CD 管線以建置、測試和部署變更。

1. 從BANKDEMO專案的內容功能表中，選擇 [小組/提交]。
2. 在「Git 暫存」索引標籤中，輸入下列提交訊息：Added Total Amount Calculation。
3. 選擇提交並推送... 。
4. 打開控 CodePipeline 制台並檢查管道執行的狀態。

> **Note**
>
> 如果您遇到企業開發人員或團隊功能提交或推送的任何問題，請使用 Git Bash 命令行界面。

練習 2：在 BankDemo申請中提取貸款計算

主題

- 第 1 步：重構貸款計算常式到一個COBOL部分
- 步驟 2：將貸款計算程序提取到獨立COBOL程序
- 步驟 3：提交變更並執行 CI/CD 管線

在下一個練習中，您將處理另一個變更請求範例。在這個案例中，貸款部門想要重複使用貸款計算常式作為獨立的 WebService。例程應該保持在COBOL，也應該仍然可以從現有CICSCOBOL程序BBANK70P.CBL調用。

第 1 步：重構貸款計算常式到一個COBOL部分

在第一步中，我們將貸款計算程序提取到一個COBOL部分。需要執行此步驟，才能在下一個步驟中將COBOL程式碼解壓縮到獨立程式中。

1. 在COBOL編輯器BBANK70P.CBL中開啟。
2. 在編輯器中，從上下文菜單中選擇「代碼分析/貸款計算更新」。這只會掃描目前的來源是否有分析規則中定義的病毒碼。
3. 在 [程式碼分析] 索引標籤的結果中，尋找第一個算術陳述式DIVIDE WS-LOAN-INTEREST BY 12。
4. 雙擊該語句以在編輯器中導航到源行。這是貸款計算程序的第一個聲明。
5. 標記下面的代碼塊，用於貸款計算例程被提取到一個部分。

```
DIVIDE WS-LOAN-INTEREST BY 12
             GIVING WS-LOAN-INTEREST ROUNDED.
           COMPUTE WS-LOAN-MONTHLY-PAYMENT ROUNDED =
             ((WS-LOAN-INTEREST * ((1 + WS-LOAN-INTEREST)
                 ** WS-LOAN-TERM)) /
             (((1 + WS-LOAN-INTEREST) * WS-LOAN-TERM) - 1 ))
```

```
                 * WS-LOAN-PRINCIPAL.
EXER01     COMPUTE WS-LOAN-TOTAL-PAYMENT =
EXER01               (WS-LOAN-MONTHLY-PAYMENT * WS-LOAN-TERM).
```

6. 從編輯器的右鍵功能表中，選擇「重構/擷取至區段...」。 。
7. 輸入新區段名稱:LOAN-CALCULATION。
8. 選擇確定。

   標記的代碼塊現在已被提取到新的LOAN-CALCULATION部分，並且代碼塊已被替換為PERFROM LOAN-CALCULATION語句。

測試變更

若要測試變更，請重複以下各節中所述的步驟。

1. 從企業開發人員啟動BANKDEMO伺服器
2. 啟動倫巴 3270 終端機
3. 執行交 BankDemo易

   此外，您現在還應該看到文本Total Loan Amount.....: $10230.60。
4. 從企業開發人員停止BANKDEMO服務器

> **Note**
>
> 如果您想避免上述步驟將代碼塊提取到一個部分，則可以將步驟 1 的修改後的源代碼從複製..\BANKDEMO-exercise\Exercis02\Step1\cobol\BBANK70P.CBL到D:\PhotonUser\workspace\bankdemo\source\cobol。

步驟 2：將貸款計算程序提取到獨立COBOL程序

在步驟 2 中，LOAN-CALCULATION部分中的代碼塊將被提取到一個獨立的程序，原始代碼將被替換為代碼來調用新的子程序。

1. BBANK70P.CBL在編輯器中開啟並尋找在步驟 1 中建立的新PERFORM LOAN-CALCULATION陳述式。
2. 將游標置於區段名稱內。它將被標記為灰色。

3. 從上下文菜單中，選擇重構-> 提取部分/段落編程... 。
4. 在「擷取要程式的段落/段落」中，輸入新檔案名稱:。LOANCALC CBL。
5. 選擇確定。

   新LOANCALC.CBL程式將在編輯器中開啟。
6. 向下捲動並檢閱為呼叫介面擷取和產生的程式碼。
7. 選擇編輯器，BBANK70P.CBL然後轉到LOAN-CALCULATION SECTION。查看正在生成的代碼以調用新的子程序LOANCALC.CBL。

> **Note**
>
> CALL語句正在使用DFHEIBLK和DFHCOMMAREALOANCALC與CICS控制塊調用。因為我們要調用新的LOANCALC.CBL子程序作為非CICS程序，我們必須刪除DFHEIBLK和DFHCOMMAREA從調用無論是通過註釋出或刪除。

測試變更

若要測試變更，請重複以下各節中所述的步驟。

1. 從企業開發人員啟動BANKDEMO伺服器
2. 啟動倫巴 3270 終端機
3. 執行交 BankDemo易

   此外，您現在還應該看到文本Total Loan Amount.....: $10230.60。
4. 從企業開發人員停止BANKDEMO服務器

> **Note**
>
> 如果要避免上述步驟將代碼塊提取到一個部分，則可以將步驟 1 的修改後源複製到..\BANKDEMO-exercise\Exercis02\Step2\cobol\BBANK70P.CBL和複製LOANCALC.CBL 到D:\PhotonUser\workspace\bankdemo\source\cobol。

步驟 3：提交變更並執行 CI/CD 管線

將變更提交至中央 CodeCommit 重新試驗，並觸發 CI/CD 管線以建置、測試和部署變更。

1. 從BANKDEMO專案的內容功能表中，選擇 [小組/提交]。
2. 在「Git 暫存」索引標籤中
   - 在未分段階段LOANCALC中新增。 CBL和LOANCALC。 CBL. MFDIR 集。
   - 輸入提交訊息：`Added Total Amount Calculation`。
3. 選擇提交並推送... 。
4. 打開控 CodePipeline 制台並檢查管道執行的狀態。

> **Note**
>
> 如果您遇到企業開發人員或團隊功能提交或推送的任何問題，請使用 Git Bash 命令行界面。

清除資源

如果您不再需要為此教學課程建立的資源，請刪除這些資源，以免繼續支付這些資源的費用。請完成下列步驟：

- 刪除 CodePipeline 配管。如需詳細資訊，請參閱《AWS CodePipeline 使用指南》 CodePipeline中的〈刪除管線〉。
- 刪除存 CodeCommit 放庫。如需詳細資訊，請參閱《AWS CodeCommit 使用指南》中的「刪除CodeCommit存放庫」。
- 刪除 S3; 儲存貯體。如需詳細資訊，請參閱 Amazon 簡單儲存服務使用者指南中的刪除儲存貯體。
- 刪除 AWS CloudFormation 堆疊。如需詳細資訊，請參閱《使用指南》中的〈刪除 AWS CloudFormation 主控台上的堆疊AWS CloudFormation〉。

教學課程：設定 AppStream 2.0 以搭配 Micro Focus 企業分析儀和微焦點企業開發人員使用

AWS 大型主機現代化透過 Amazon 2.0 提供多種工具。 AppStream AppStream 2.0 是全受管、安全的應用程式串流服務，可讓您將桌面應用程式串流給使用者，而無需重新撰寫應用程式。 AppStream 2.0

為使用者提供即時存取所需的應用程式，並在他們選擇的裝置上提供反應靈敏、流暢的使用者體驗。使用 AppStream 2.0 託管執行階段引擎特定工具，可讓客戶應用程式團隊直接從 Web 瀏覽器使用這些工具，與存放在 Amazon S3 儲存貯體或 CodeCommit儲存庫中的應用程式檔案互動。

如需 2.0 版瀏覽器 Support 的相關資訊，請參閱《Amazon AppStream AppStream 2.0 管理指南》中的系統需求和功能支援 (網頁瀏覽器)。如果您在使用 AppStream 2.0 時遇到問題，請參閱《Amazon AppStream 2.0 管理指南》中的 AppStream 2.0 使用者問題疑難排解。

本文件適用於客戶營運團隊的成員。本文說明如何設定 Amazon AppStream 2.0 叢集和堆疊，以託管 Micro Focus 企業分析器和 Micro Focus 企業開發人員工具與 AWS 大型主機現代化搭配使用。Micro Focus 企業分析器通常在「評估」階段使用，而 Micro Focus 企業開發人員通常會在 AWS 大型主機現代化方法的「移轉和現代化」階段使用。如果您打算同時使用 Enterprise Analyzer 和企業開發人員，您必須為每個工具建立不同的叢集和堆疊。每個工具都需要自己的叢集和堆疊，因為它們的授權條款不同。

> **Important**
>
> 本教程中的步驟基於可下載的 AWS CloudFormation 模板 cfn-m2-appstream-fleet-ea-ed .yml。

主題

- 必要條件
- 第 1 步：獲取 AppStream 2.0 圖像
- 步驟 2：使用 AWS CloudFormation 範本建立堆疊
- 步驟 3：在 AppStream 2.0 中建立使用者
- 步驟四：登入 AppStream 2.0
- 步驟 5：在 Amazon S3 中驗證存儲桶（可選）
- 後續步驟
- 清除資源

必要條件

- 下載範本：CFN-平方米-appstream-fleet-ea-ed .yml。

- 取得預設群組VPC和安全性群組的 ID。如需有關預設值的詳細資訊VPC，請參閱 Amazon VPC 使用者指南VPCs中的預設值。如需有關預設安全群組的詳細資訊，請參閱 Amazon EC2 使用者指南中的預設和自訂安全群組。
- 確保您具有以下權限：
  - 在 AppStream 2.0 中建立堆疊、叢集和使用者。
  - AWS CloudFormation 使用範本建立堆疊。
  - 建立儲存貯體並將檔案上傳到 Amazon S3 中的儲存貯體。
  - 從下載憑證（`access_key_id`和`secret_access_key`）IAM。

第 1 步：獲取 AppStream 2.0 圖像

在此步驟中，您會與您的 AWS 帳戶共用企業分析器和企業開發人員的 AppStream 2.0 映像。

1. 在開啟大 AWS 型主機現代化主控台。https://console.aws.amazon.com/m2/
2. 在左側導覽中，選擇 [工具]。
3. 在 [分析、開發和建置資產] 中，選擇 [與我的 AWS 帳戶共用資產]。

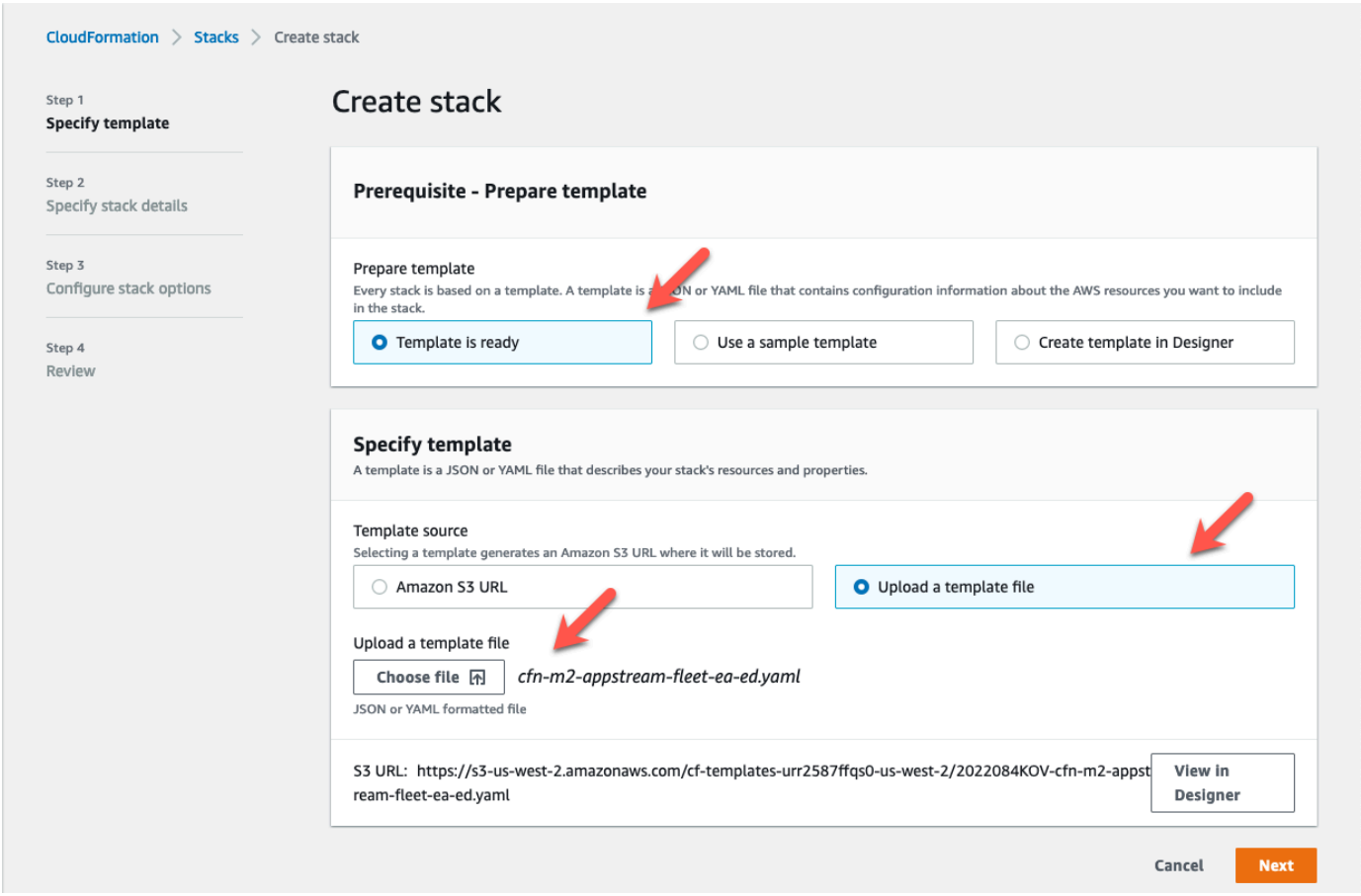
步驟 2：使用 AWS CloudFormation 範本建立堆疊

在此步驟中，您可以使用下載的 AWS CloudFormation 範本建立 AppStream 2.0 堆疊和叢集，以執行 Micro Focus 企業分析器。您可以稍後重複此步驟，建立另一個 AppStream 2.0 堆疊和叢集來執行 Micro Focus 企業開發人員，因為每個工具都需要在 AppStream 2.0 中自己的叢集和堆疊。如需 AWS CloudFormation 堆疊的詳細資訊，請參閱使用指南中的AWS CloudFormation 使用堆疊。

> **Note**
>
> AWS 大型主機現代化為使用企業分析儀和企業開發人員的標準 AppStream 2.0 定價增加額外費用。如需詳細資訊，請參閱AWS 大型主機現代化定價。

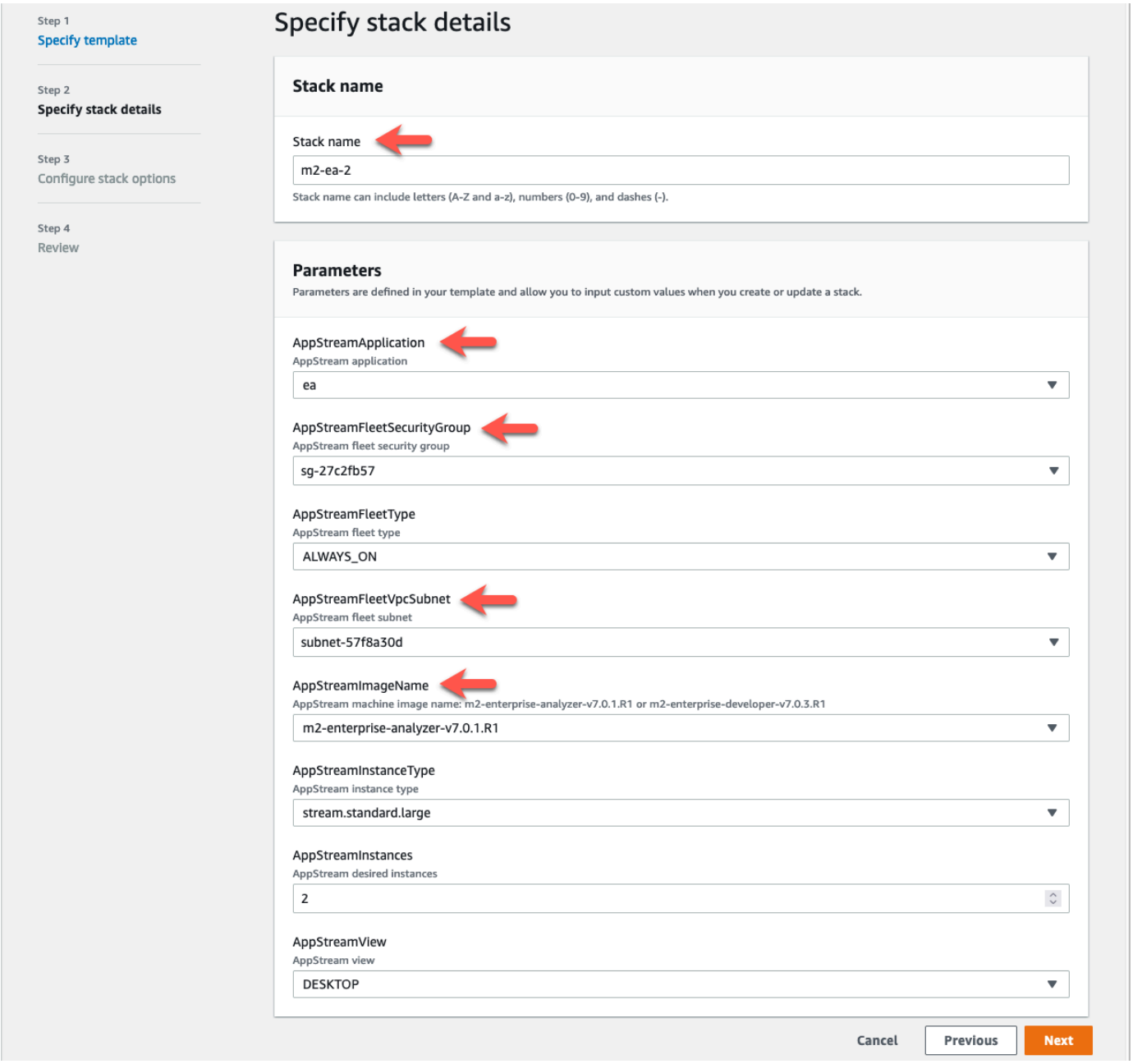
1. 如有必要，請下載 cfn-m2-appstream-fleet-ea-ed .yml 模板。
2. 開啟主 AWS CloudFormation 控台並選擇 [建立堆疊]，並使用新資源 (標準)。
3. 在先決條件-準備範本中，選擇範本已準備就緒。
4. 在「指定範本」中，選擇「上傳範本檔案」。
5. 在 [上傳範本檔案] 中，選擇 [選擇檔案] 並上傳 cfn-m2-appstream-fleet-ea-ed. yml 範本。

6. 選擇 Next (下一步)。



7. 在指定堆疊詳細資料上，輸入下列資訊：

   - 在堆疊名稱中，輸入您選擇的名稱。例如：**`m2-ea`**。
   - 在中 AppStreamApplication，選擇 ea。
   - 在中 AppStreamFleetSecurityGroup，選擇預設VPC的預設安全性群組。
   - 在中 AppStreamFleetVpcSubnet，選擇預設值內的子網路VPC。
   - 在中 AppStreamImageName，選擇開頭為的影像`m2-enterprise-analyzer`。此影像包含目前支援的 Micro Focus 企業分析器工具版本。
   - 接受其他欄位的預設值，然後選擇「下一步」。



8. 接受所有預設值，然後再次選擇下一步。
9. 在 [檢閱] 中，請確定所有參數都符合您的需求。
10. 捲動至底部，選擇 [我確認AWS CloudFormation 可能會使用自訂名稱建立IAM資源]，然後選擇 [建立堆疊]。

建立堆疊和叢集需要 20 到 30 分鐘。您可以選擇「重新整理」以查看 AWS CloudFormation 事件發生時的事件。

步驟 3：在 AppStream 2.0 中建立使用者

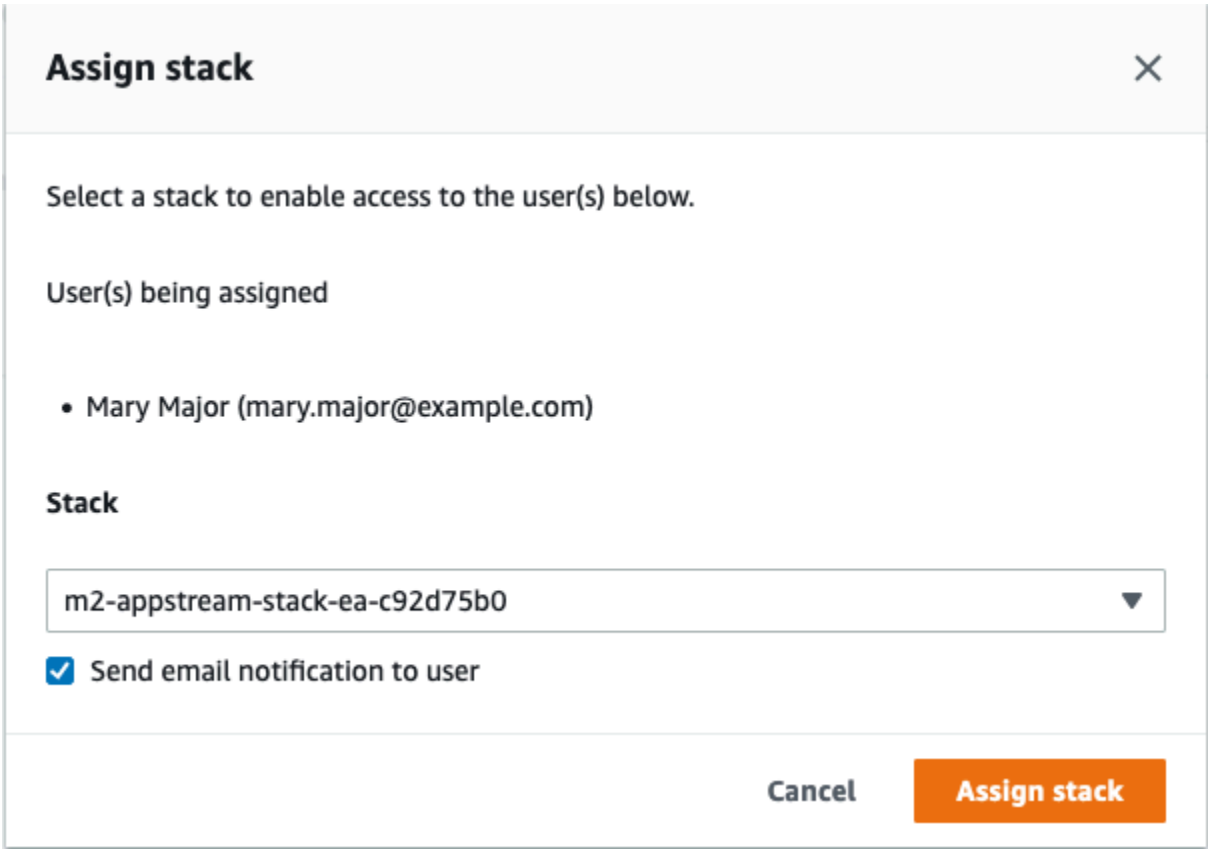
等待 AWS CloudFormation 完成堆疊建立時，您可以在 AppStream 2.0 中建立一或多個使用者。這些用戶是那些誰將在 AppStream 2.0 使用企業分析器。您需要為每個使用者指定一個電子郵件地址，並確保每個使用者都有足夠的許可在 Amazon S3 中建立儲存貯體、將檔案上傳到儲存貯體，以及連結至儲存貯體以對應其內容。

1. 開啟 AppStream 2.0 主控台。
2. 在左側導覽列中，選擇 [使用者集區]。
3. 選擇 Create user (建立使用者)。
4. 提供電子郵件地址，讓使用者可以接收使用 AppStream 2.0 的電子郵件邀請、名字和姓氏，然後選擇 [建立使用者]。
5. 如有必要，請重複以上步驟以建立更多 每個使用者的電子郵件地址必須是唯一的。

如需有關建立 AppStream 2.0 使用者的詳細資訊，請參閱《Amazon AppStream 2.0 管理指南》中的 AppStream 2.0 個使用者集區。

建立堆疊 AWS CloudFormation 完成後，您可以將您建立的使用者指派給堆疊，如下所示：

1. 開啟 AppStream 2.0 主控台。
2. 選擇使用者名稱。
3. 選擇動作，然後選擇指派堆疊。
4. 在指派堆疊中，選擇開頭為的堆疊`m2-appstream-stack-ea`。
5. 選擇 Assign stack (指派堆疊)。



將使用者指派給堆疊會造成 AppStream 2.0 以您提供的地址傳送電子郵件給使用者。此電子郵件包含 AppStream 2.0 登入頁面的連結。

步驟四：登入 AppStream 2.0

在此步驟中，您可以使用 AppStream 2.0 傳送給您建立之使用者的電子郵件中的連結登入 AppStream 2.0 步驟 3：在 AppStream 2.0 中建立使用者。

1. 使用 AppStream 2.0 發送的電子郵件中提供的鏈接登錄到 AppStream 2.0。
2. 如果出現提示，請變更您的密碼。您看到的 AppStream 2.0 畫面類似下列內容：

3. 選擇「桌面」。
4. 在工作列上，選擇 [搜尋] 並輸入**D:**以瀏覽至 [主資料夾]。

> **Note**
>
> 如果跳過此步驟，當您嘗試存取主資料夾時，可能會收到`Device not ready`錯誤訊息。

在任何時候，如果您在登入 AppStream 2.0 時遇到問題，您可以重新啟動 AppStream 2.0 叢集，然後嘗試再次登入，使用下列步驟。

1. 開啟 AppStream 2.0 主控台。
2. 在左側導覽列中，選擇 [艦隊]。
3. 選擇您要使用的艦隊。
4. 選擇動作，然後選擇停止。
5. 等待艦隊停止。
6. 選擇動作，然後選擇開始。

此過程可能需要大約 10 分鐘。

步驟 5：在 Amazon S3 中驗證存儲桶（可選）

您用來建立堆疊的 AWS CloudFormation 範本完成的其中一項任務是在 Amazon S3 中建立兩個儲存貯體，這對於跨工作階段儲存和還原使用者資料和應用程式設定是必要的。這些桶如下所示：

- 名稱開頭為`appstream2-`。此值區會在 AppStream 2.0 (`D:\PhotonUser\My Files\Home Folder`) 中將資料對應到您的主資料夾。

> **Note**
>
> 主資料夾對於指定的電子郵件地址而言是唯一的，且會在指定 AWS 帳戶中的所有叢集和堆疊之間共用。主資料夾的名稱是使用者電子郵件地址的SHA256雜湊值，並儲存在以該雜湊為基礎的路徑上。

- 名稱開頭為`appstream-app-settings-`。此值區包含 AppStream 2.0 的使用者工作階段資訊，並包含瀏覽器我的最愛、IDE應用程式連線設定檔以及 UI 自訂等設定。如需詳細資訊，請參閱《Amazon AppStream 2.0 管理指南》中的應用程式設定持續性運作方式。

若要確認值區是否已建立，請依照下列步驟執行：

1. 開啟 Amazon S3 主控台。
2. 在左側導覽中，選擇 [值區]。
3. 在依名稱尋找值區中，輸**`appstream`**入以篩選清單。

如果您看到值區，則不需要執行進一步的動作。只要注意，存在桶。如果您沒有看到值區，表示 AWS CloudFormation 範本未完成執行，或發生錯誤。前往 AWS CloudFormation 主控台並檢閱堆疊建立訊息。

後續步驟

現在已設定 AppStream 2.0 基礎結構，您可以設定並開始使用企業分析器。如需詳細資訊，請參閱教學課程：在 AppStream 2.0 上設定企業分析器。您也可以設定企業開發人員。如需詳細資訊，請參閱教學課程：在 AppStream 2.0 上設定微焦點企業開發人員。

清除資源

清理建立的堆疊和叢集的程序，請參閱建立 AppStream 2.0 叢集和堆疊。

刪除 AppStream 2.0 物件後，帳戶管理員也可以視需要清理應用程式設定和主資料夾的 Amazon S3 儲存貯體。

> **Note**
>
> 特定使用者的主資料夾在所有叢集中都是唯一的，因此如果同一帳戶中的其他 AppStream 2.0 堆疊處於作用中狀態，您可能需要保留該資料夾。

最後， AppStream 2.0 目前不允許您使用控制台刪除用戶。相反地，您必須搭配使API用該服務CLI。如需詳細資訊，請參閱 Amazon AppStream 2.0 管理指南中的使用者集區管理。

教學課程：與微焦點企業開發人員搭配使用

本教學課程說明如何使用範本和預先定義的專案與 Micro Focus 企業開發人員。它涵蓋了三個用例。所有使用案例都使用範例中提供的 BankDemo 範例程式碼。若要下載範例，請選擇`bankdemo.zip`。

> **Important**
>
> 如果您使用適用於 Windows 的企業開發人員版本，編譯器生成的二進製文件只能在企業開發人員提供的企業服務器上運行。您無法在以 Linux 為基 AWS 礎的大型主機現代化執行階段下執行它們。

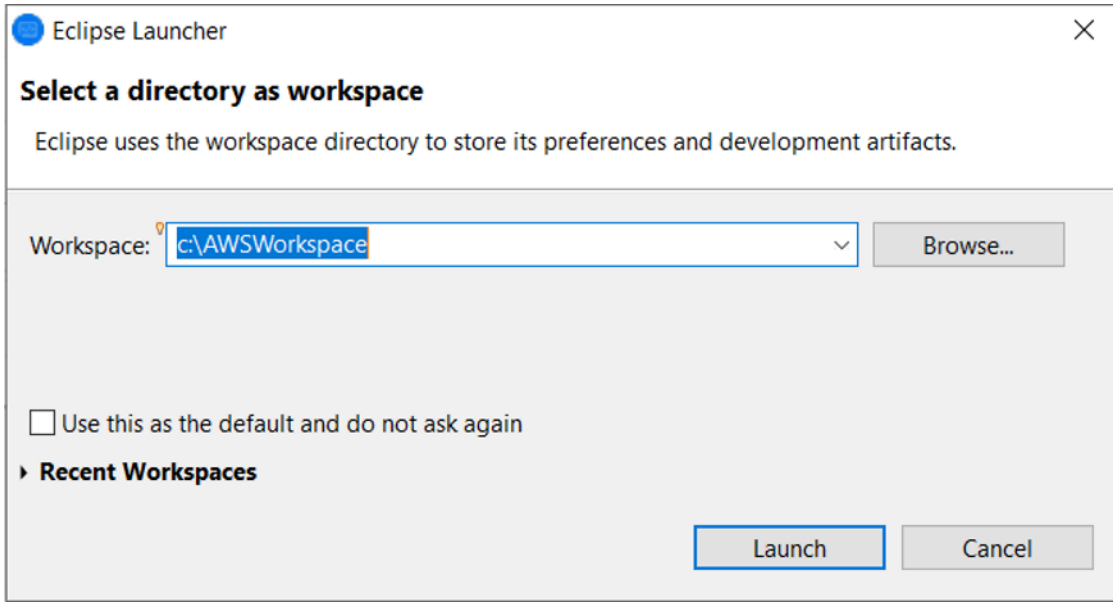
主題

- 使用案例 1-使用包含來源元件的COBOL專案範本
- 使用案例 2-使用不含來源元件的COBOL專案範本
- 使用案例 3-使用預先定義的COBOL專案連結至來源資料夾
- 使用區域定義JSON樣版

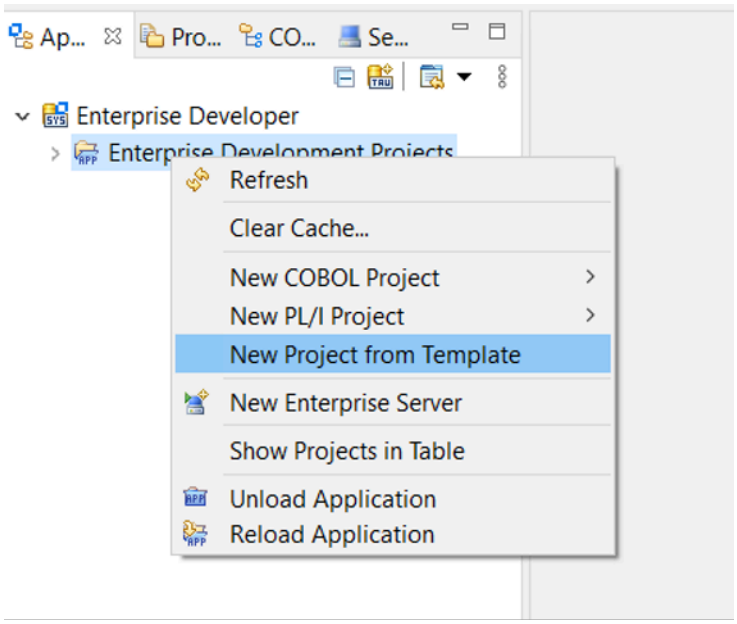
使用案例 1-使用包含來源元件的COBOL專案範本

此使用案例需要您將來源元件複製到 Template 目錄結構中，作為示範預先設定步驟的一部分。在`bankdemo.zip`這已經從原始`AWSTemplates.zip`交付中改變了，以避免有兩個源副本。

1. 啟動企業開發人員並指定選擇的工作區。

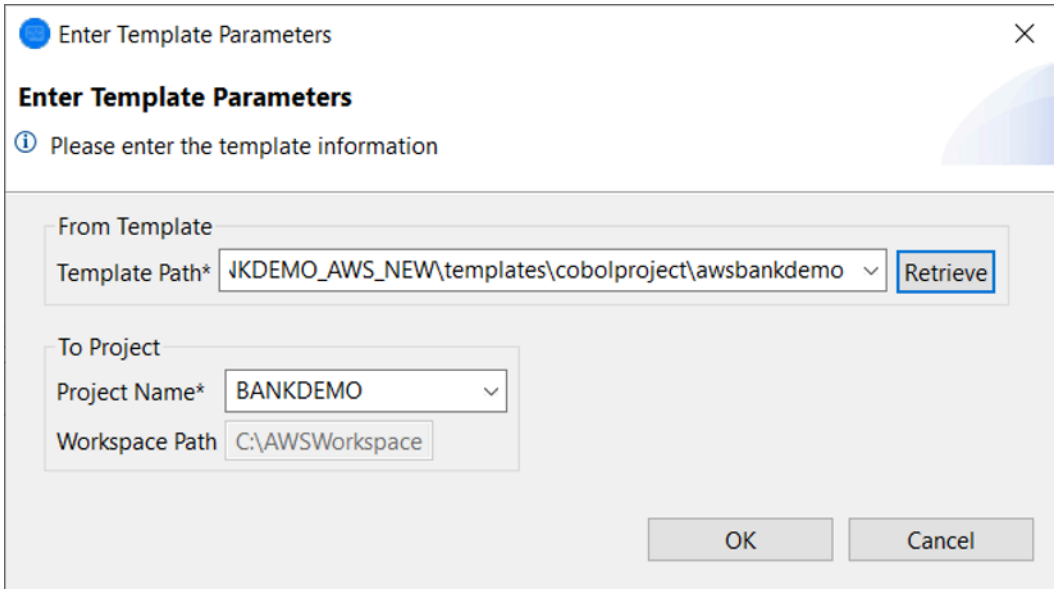


2. 在 [應用程式總管] 檢視中，從 [企業開發專案] 樹狀檢視項目，從內容功能表中選擇 [從範本新增專案]。

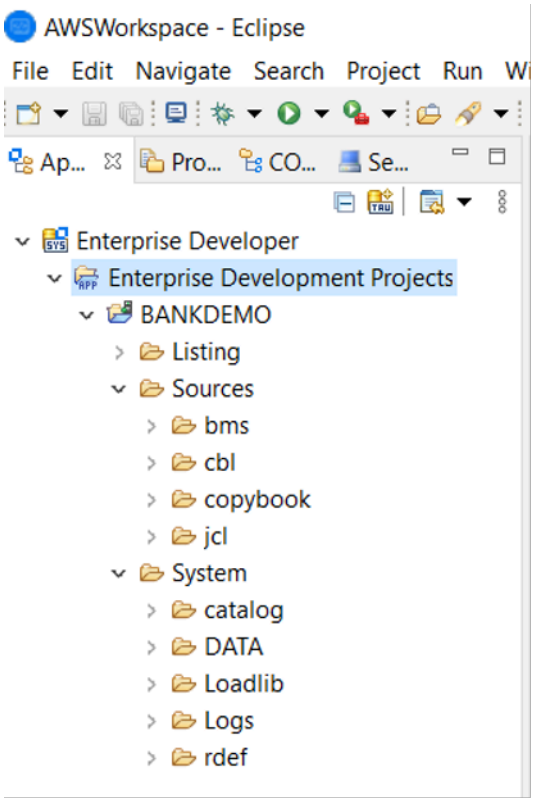


3. 輸入範本參數，如圖所示。

> **Note**
>
> 模板路徑將指的ZIP是提取的位置。



4. 選擇確定將根據提供的模板創建一個本地開發 Eclipse 項目，並具有完整的源代碼和執行環境結構。



結System構包含一個完整的資源定義檔案，其中包含的必要項目BANKDEMO、已新增項目的所需目錄以及對應的資ASCII料檔案。

因為來源範本結構包含所有來源項目，這些檔案會複製到本機專案，因此會在企業開發人員中自動建置。

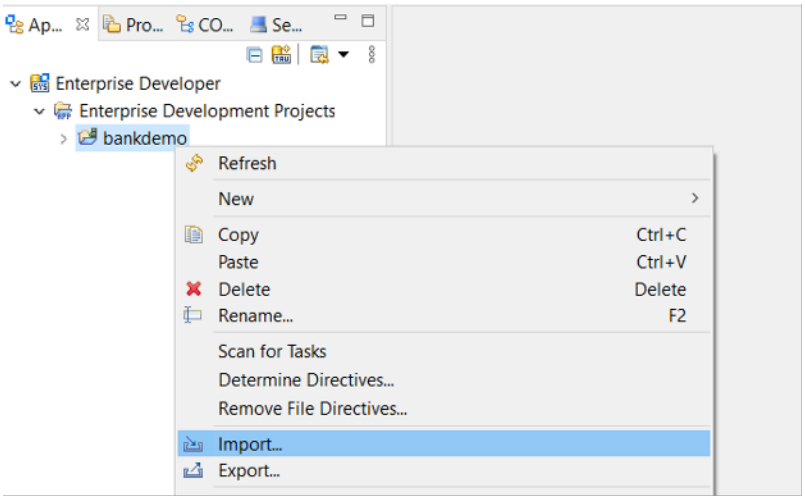
使用案例 2-使用不含來源元件的COBOL專案範本

步驟 1 到 3 是相同的使用案例 1-使用包含來源元件的COBOL專案範本。

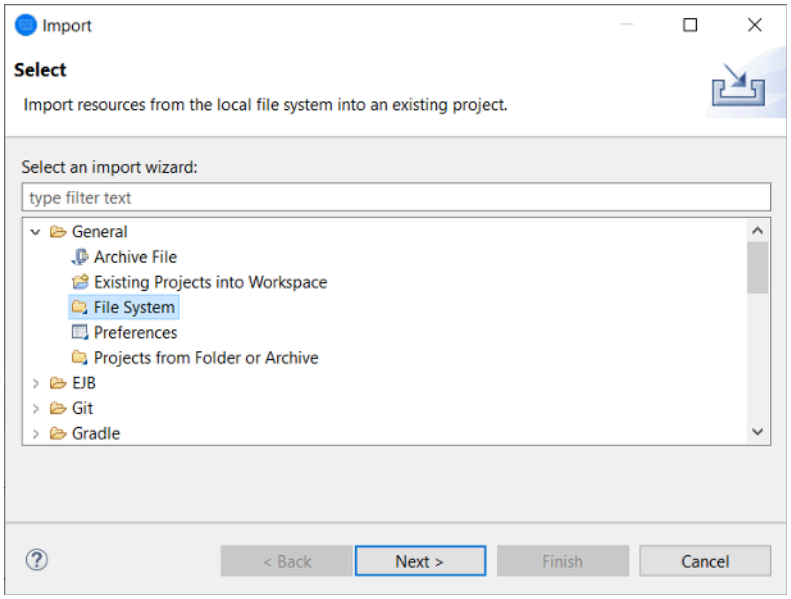
此使用案例中的System結構也包含一個完整的資源定義檔案，其中包含必要項目 BankDemo、已新增項目的所需目錄，以及對應的資ASCII料檔案。

但是，範本來源結構不包含任何元件。您必須從您正在使用的任何源存儲庫中將它們導入到項目中。

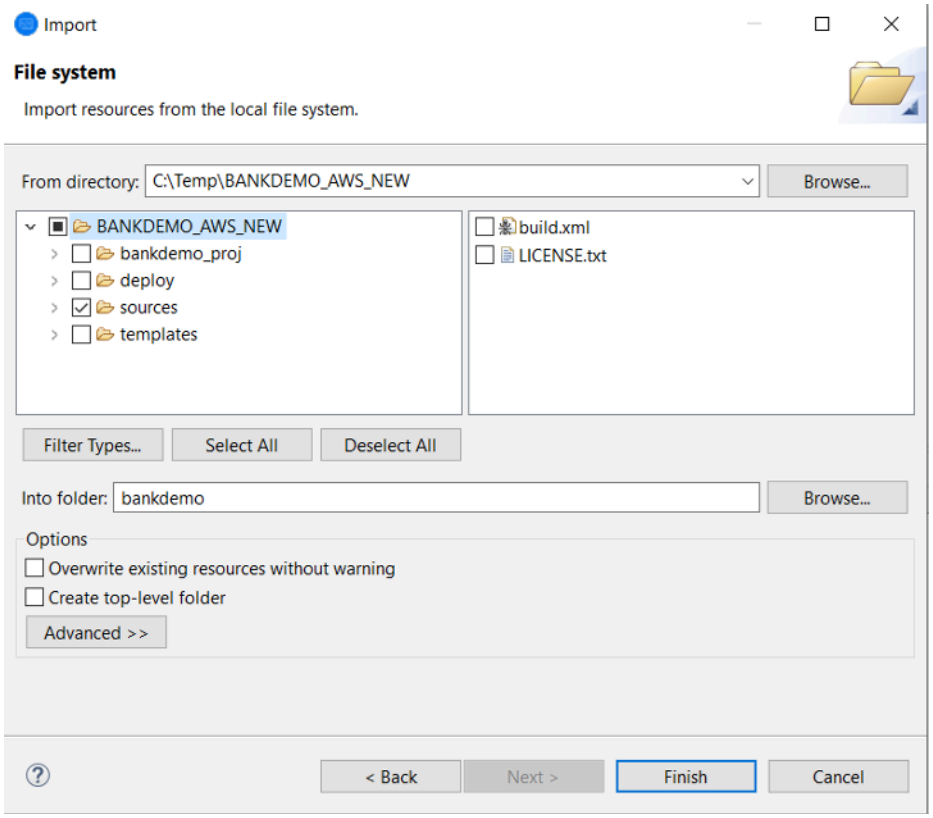
1. 選擇專案名稱。從相關內容功能表中，選擇「匯入」。



2. 在出現的對話方塊中的 [一般] 區段下，選擇 [檔案系統]，然後選擇 [下一步]。



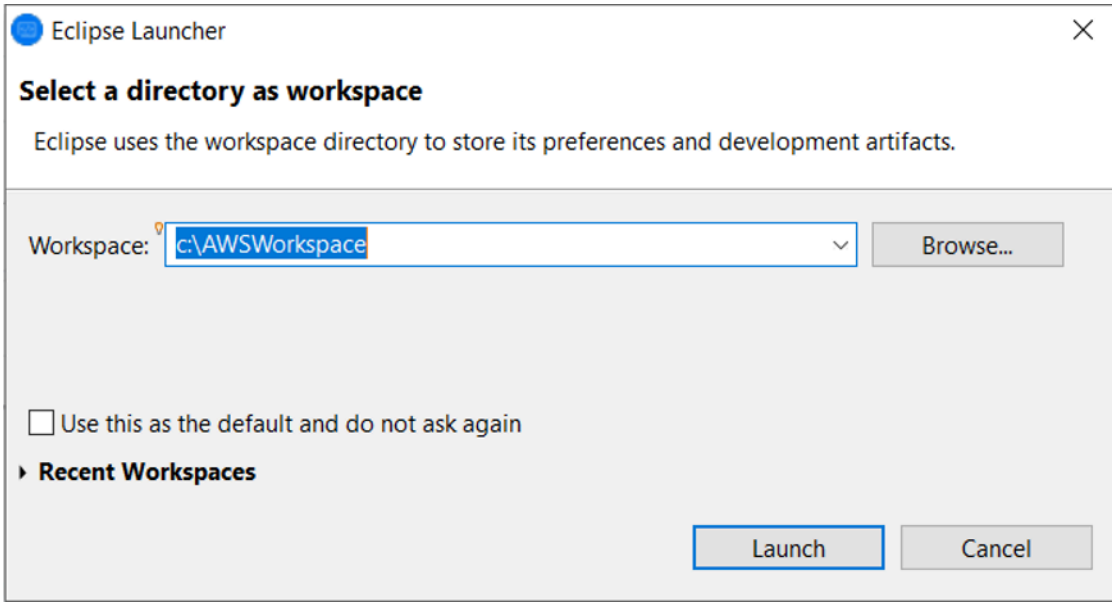
3. 瀏覽檔案系統以指向存放庫資料夾，以填入「從目錄」欄位。選擇您要導入的所有文件夾，例如`sources`。此`Into folder`欄位將會預先填入。選擇 Finish (完成)。



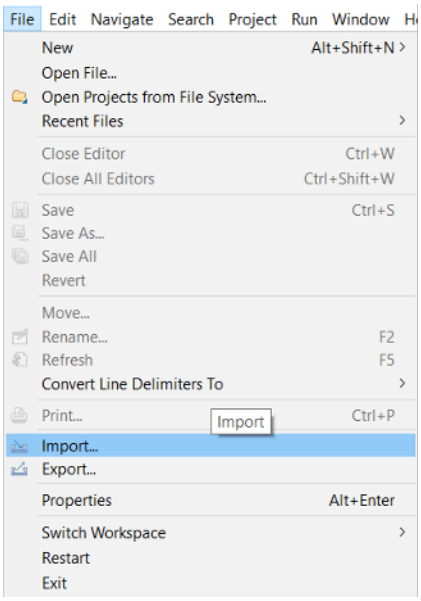
來源範本結構包含所有來源項目之後，它們會在企業開發人員中自動建置。

使用案例 3-使用預先定義的COBOL專案連結至來源資料夾

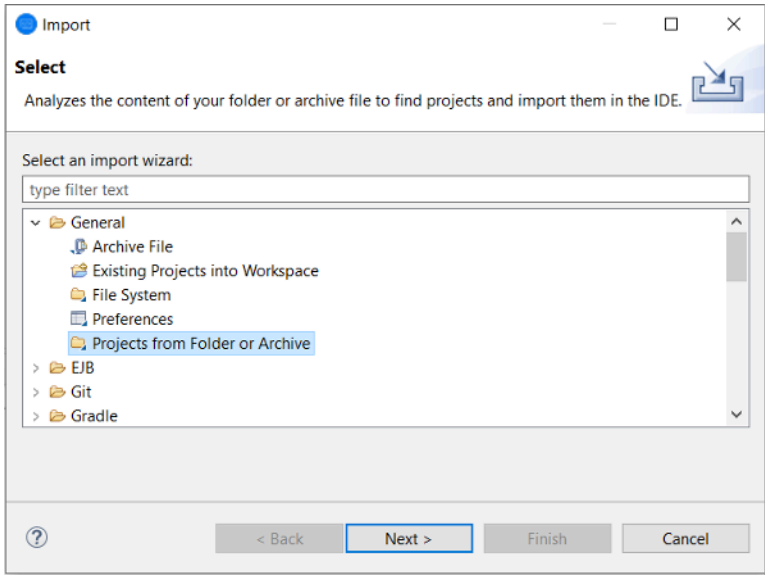
1. 啟動企業開發人員並指定選擇的工作區。



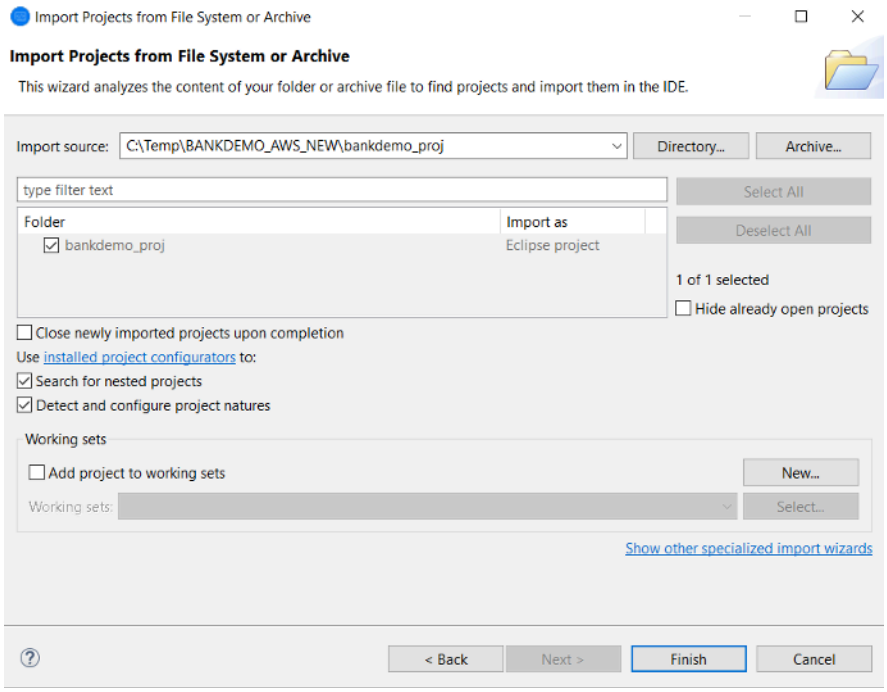
2. 從 File (檔案) 功能表中，選擇 Import (匯入)。



3. 在出現的對話框中，在「常規」下，選擇「文件夾中的項目」或「存檔」，然後選擇



4. 填入匯入來源、選擇目錄並瀏覽檔案系統以選取預先定義的專案資料夾。其中包含的專案具有指向同一儲存庫中來源資料夾的連結。

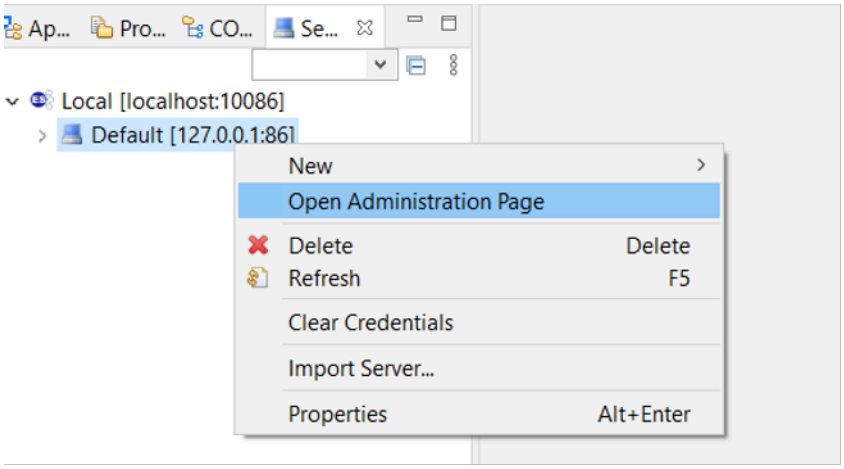


選擇 Finish (完成)。

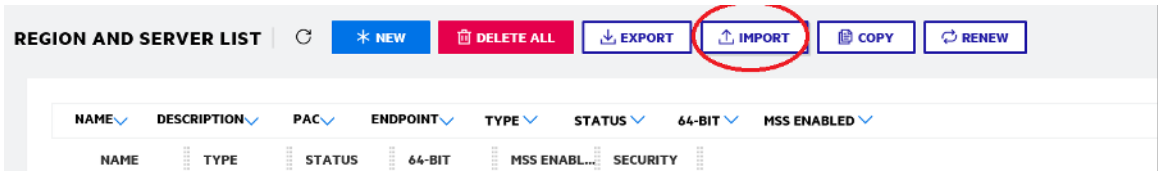
由於專案是由來源資料夾的連結填入，因此會自動建置程式碼。

使用區域定義JSON樣版

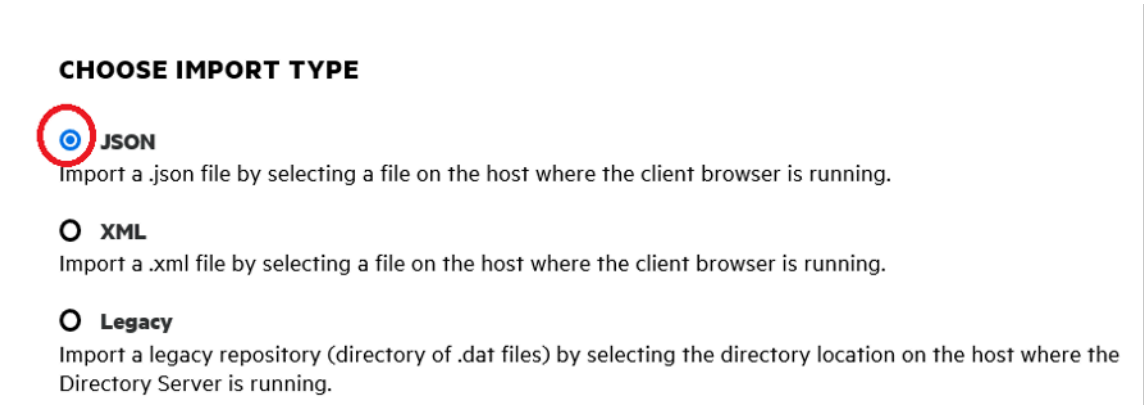
1. 切換到伺服器資源管理器視圖。從相關內容功能表中，選擇「開啟管理頁面」，以啟動預設瀏覽器。



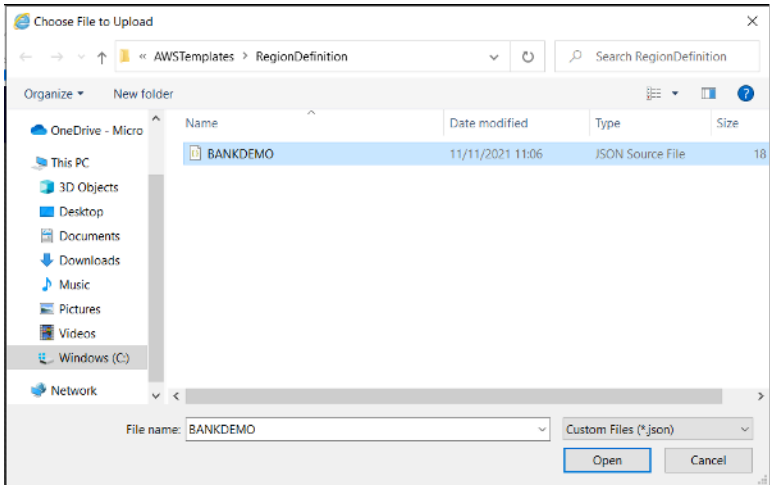
2. 在產生的「企業伺服器通用 Web 管理」(ESCWA) 畫面中，選擇「匯入」。



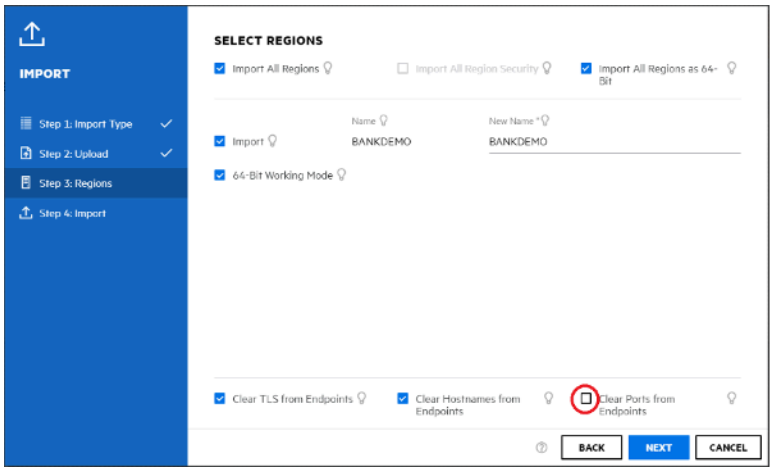
3. 選擇JSON匯入類型，然後選擇 [下一步]。



4. 上傳提供的BANKDEMO.JSON文件。

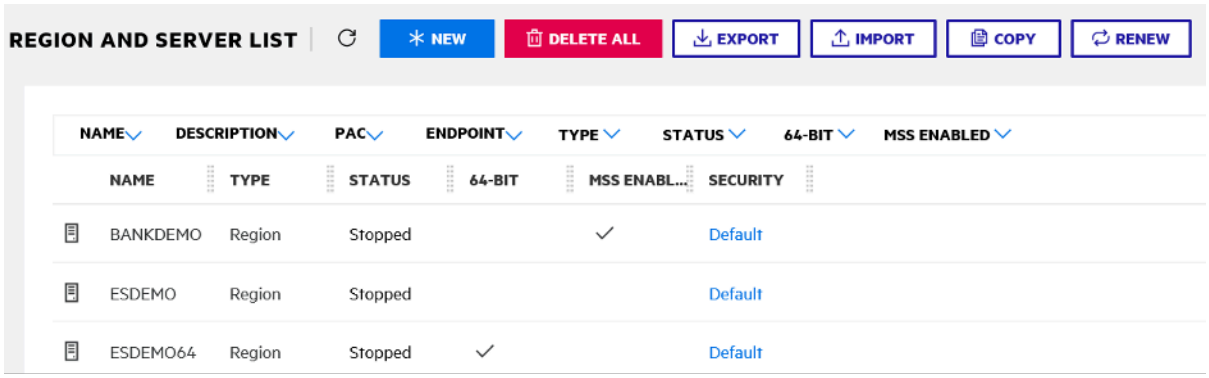


選取之後，選擇 [下一步]。



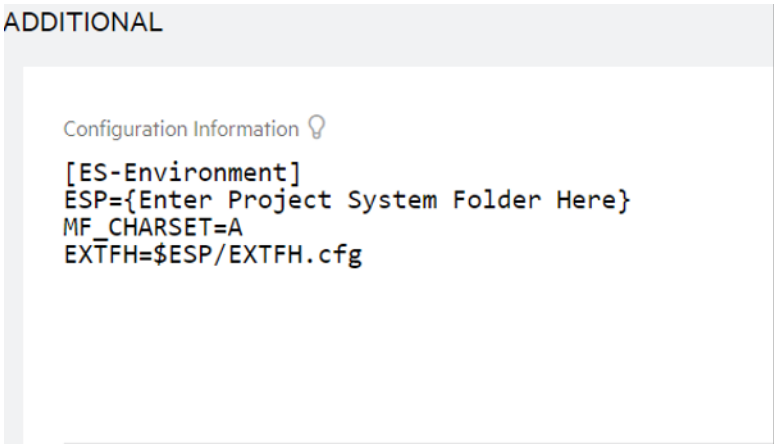
在「選取區域」面板上，確定未選取「從端點清除連接埠」選項，然後繼續透過面板選擇「下一步」，直到顯示「執行匯入」面板為止。然後選擇進口從左側導航窗格中。

最後點擊完成。然後，該BANKDEMO區域將被添加到服務器列表中。



5. 切換作業選項至BANKDEMO區域的「一般屬性」。

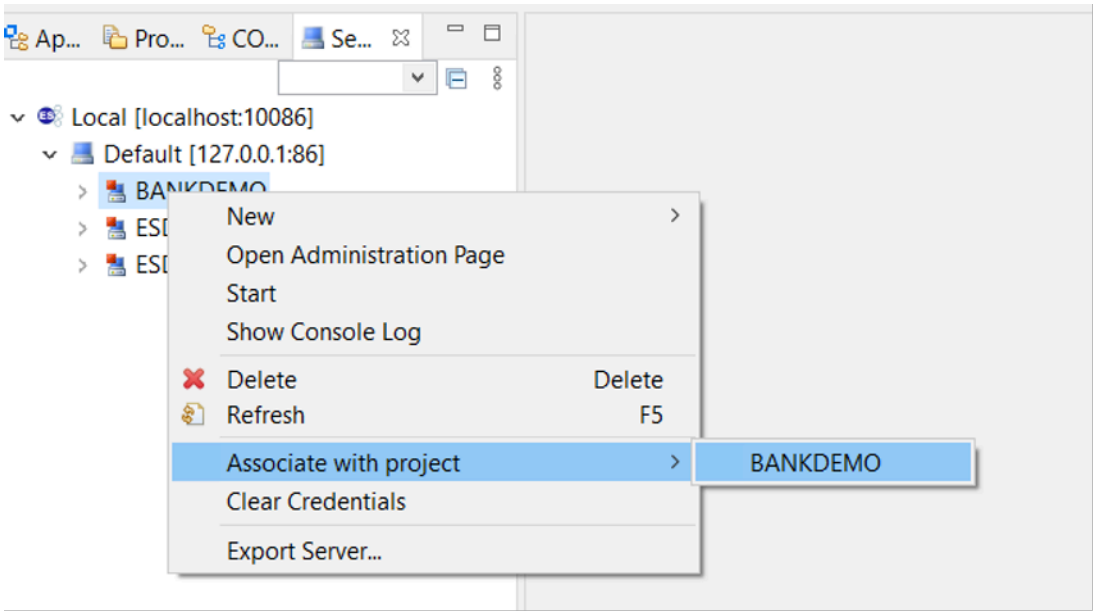
6. 捲動至組態區段。
7. ESP環境變量需要被設置為與在前面的步驟中創建的 Eclipse 項目相關的`System`文件夾。這應該是`workspacefolder/projectname/System`。



8. 按一下 Apply (套用)。

   該區域現在已完全配置為與 Eclipse COBOL 項目一起運行。

9. 最後，回到企業開發人員，將匯入的區域與專案相關聯。



企業開發人員環境現在已準備就緒，可以使用完整的工作版本 BankDemo。您可以針對區域編輯、編譯和偵錯程式碼。

> **Important**
>
> 如果您使用適用於 Windows 的企業開發人員版本，編譯器生成的二進製文件只能在企業開發人員提供的企業服務器上運行。您無法在以 Linux 為基 AWS 礎的大型主機現代化執行階段下執行它們。

教學課程：在 AppStream 2.0 上設定企業分析器

本教學課程說明如何設定 Micro Focus 企業分析器來分析一或多個大型主機應用程式。Enterprise Analyzer 工具會根據應用程式原始程式碼和系統定義的分析，提供數個報告。

此設置旨在促進團隊協作。安裝使用 Amazon S3 儲存貯體與虛擬磁碟共用原始程式碼。這樣做可以在 Windows 機器上使用複製器。透過RDS執行 Postgre 的通用 Amazon 執行個體SQL，任何團隊成員都可以存取所有要求的報告。

團隊成員也可以在其個人機器上掛載虛擬 Amazon S3 支援的磁碟。並從工作站更新來源儲存貯體。如果連線至其他內部部署內部系統，他們可能會在其電腦上使用指令碼或任何其他形式的自動化。

此設定是以大型主機現代化與客戶共用的 AppStream 2.0 Windows 映像為基 AWS 礎。安裝程式也以 AppStream 2.0 叢集和堆疊的建立為基礎，如中教學課程：設定 AppStream 2.0 以搭配 Micro Focus 企業分析儀和微焦點企業開發人員使用所述。

> **Important**
>
> 本教學課程中的步驟假設您使用可下載的 AWS CloudFormation 範本 cfn-m2-appstream-fleet-ea-ed .yml 設定 AppStream 2.0。如需詳細資訊，請參閱教學課程：設定 AppStream 2.0 以搭配 Micro Focus 企業分析儀和微焦點企業開發人員使用。
> 若要執行本教學課程中的步驟，您必須已設定 Enterprise Analyzer 叢集和堆疊，而且它們必須在執行中。

如需企業分析器功能和交付項目的完整說明，請參閱 Micro Focus 網站上的企業分析器文件。

圖片內容

除了企業分析器應用程式本身之外，影像還包含下列工具和程式庫。

第三方工具

- Python
- 複製品
- pgAdmin
- 供應鏈管理
- 後驅動程序 SQL ODBC

中的圖書館 `C:\Users\Public`

- BankDemo 企業開發人員的源代碼和項目定義：`m2-bankdemo-template.zip`。
- MFA大型主機的安裝套件：。`mfa.zip`如需詳細資訊，請參閱 Micro Focus 企業開發人員文件中的大型主機存取概觀。
- Rclone 的命令和配置文件（在教程中使用它們的說明）：`m2-rclone.cmd`和`m2-rclone.conf`。

主題

- 必要條件
- 步驟 1：設定
- 步驟 2：在視窗上建立基於 Amazon S3 的虛擬資料夾
- 步驟 3：為 Amazon RDS 執行個體建立ODBC來源
- 後續會話
- 工作區連線疑難
- 清除資源

必要條件

- 將您要分析的客戶應用程式的原始程式碼和系統定義上傳到 S3 儲存貯體。系統定義CICSCSD包括DB2物件定義等。您可以在值區內建立資料夾結構，以便組織應用程式成品的方式。例如，當您解壓縮 BankDemo 樣本時，它具有以下結構：

```
demo
     |--> jcl
     |--> RDEF
     |--> transaction
     |--> xa
```

- 創建並啟動一個運行 Postgre SQL 的 Amazon RDS 實例。這個執行個體會儲存企業分析器所產生的資料和結果。您可以與應用程式小組的所有成員共用此執行個體。此外，在數據庫中創建一個名為m2_ea（或任何其他合適的名稱）的空模式。定義授權使用者的認證，讓他們建立、插入、更新和刪除此結構描述中的項目。您可以從 Amazon RDS 主控台或帳戶管理員取得資料庫名稱URL、其伺服器端點和TCP連接埠。
- 請確定您已設定程式設計存取您 AWS 帳戶的. 如需詳細資訊，請參閱中的Amazon Web Services 一般參考程式設計存取。

步驟 1：設定

1. 使用您在 AppStream 2.0 的歡迎電子郵件訊息中收到的 AppStream 2.0 開始工作階段。URL
2. 使用您的電子郵件作為用戶 ID，並定義您的永久密碼。
3. 選取企業分析器堆疊。
4. 在 AppStream 2.0 功能表頁面上，選擇 [桌面] 以到達叢集正在串流處理的 Windows 桌面平台。

步驟 2：在視窗上建立基於 Amazon S3 的虛擬資料夾

> **Note**
>
> 如果您已在 AWS 大型主機現代化預覽期間使用 Rclone，則必須更新m2-rclone.cmd至中的較新版本。C:\Users\Public

1. C:\Users\PhotonUser\My Files\Home Folder使用 [檔m2-rclone.cmd案總管] 將中提供的m2-rclone.conf和檔案複製C:\Users\Public到您的個人資料夾。
2. 使用m2-rclone.conf您的 AWS 訪問密鑰和相應的密鑰以及您的 AWS 區域.

```
[m2-s3]
type = s3
provider = AWS
access_key_id = YOUR-ACCESS-KEY
secret_access_key = YOUR-SECRET-KEY
region = YOUR-REGION
acl = private
server_side_encryption = AES256
```

3. 在 m2-rclone.cmd 中，進行下列變更：

- 更改`your-s3-bucket`為您的 Amazon S3 存儲桶名稱。例如：`m2-s3-mybucket`。
- 變更`your-s3-folder-key`為您的 Amazon S3 儲存貯體金鑰。例如：`myProject`。
- 變更`your-local-folder-path`至您要從包含這些檔案的 Amazon S3 儲存貯體同步應用程式檔案的目錄路徑。例如：`D:\PhotonUser\My Files\Home Folder\m2-new`。此同步的目錄必須是主資料夾的子目錄， AppStream 2.0 才能在作業階段開始和結束時正確備份和還原它。

```
:loop
timeout /T 10
"C:\Program Files\rclone\rclone.exe" sync m2-s3:your-s3-bucket/your-s3-folder-
key "D:\PhotonUser\My Files\Home Folder\your-local-folder-path" --config "D:
\PhotonUser\My Files\Home Folder\m2-rclone.conf"
goto :loop
```

4. 打開一個 Windows 命令提示符，如`C:\Users\PhotonUser\My Files\Home Folder`果需要，請點擊並運行`m2-rclone.cmd`。此命令指令碼會執行連續迴圈，每 10 秒將 Amazon S3 儲存貯體和金鑰同步到本機資料夾。您可以根據需要調整逾時。您應該會看到位於 Windows 檔案總管中 Amazon S3 儲存貯體中的應用程式原始程式碼。

若要將新檔案新增至您正在處理的檔案集或更新現有檔案，請將檔案上傳到 Amazon S3 儲存貯體，這些檔案會在定義的下一次迭代時同步到您的目錄`m2-rclone.cmd`。同樣地，如果您想刪除某些檔案，請從 Amazon S3 儲存貯體中刪除這些檔案。下一個同步操作將從您的本地目錄中刪除它們。

步驟 3：為 Amazon RDS 執行個體建立ODBC來源

1. 若要啟動 EA_Admin 工具，請瀏覽至瀏覽器視窗左上角的應用程式選取器功能表，然後選擇 MF EA_Admin。
2. 從「管理」功能表中選擇「ODBC資料來源」，然後從 DSN「使用者」標籤選擇「新增」。
3. 在 [建立新資料來源] 對話方塊中，選擇 [移除 SQLUnicode 驅動程式]，然後選擇 [完成]。
4. 在「移除 SQLUnicode ODBC 驅動程式 (psqlODBC) 設定」對話方塊中，定義並記下您想要的資料來源名稱。使用先前建立之RDS例證的值完成下列參數：

   描述

   選用說明可協助您快速識別此資料庫連線。

資料庫

您之前創建的 Amazon RDS 數據庫。

Server

Amazon RDS 端點。

連線埠

Amazon RDS 港口。

使用者名稱

如 Amazon RDS 執行個體所定義。

密碼

如 Amazon RDS 執行個體所定義。

5. 選擇「測試」以驗證與 Amazon 的連線RDS是否成功，然後選擇「儲存」以儲存新使用者DSN。
6. 等到您看到確認正確工作區建立的訊息，然後選擇 [確定] 以完成 [ODBC資料來源] 並關閉 EA_Admin 工具。
7. 再次瀏覽至應用程式選取器功能表，然後選擇「企業分析器」以啟動工具。選擇「新建」。
8. 在「工作區」組態視窗中，輸入您的工作區名稱並定義其位置。如果您在此組態下工作，工作區可以是 Amazon S3 磁碟，或者您的主資料夾 (如果您願意)。
9. 選擇「選擇其他資料庫」以連接到您的 Amazon RDS 執行個體。
10. 從選項中選擇下一個圖標，然後選擇 OK。
11. 對於 [選項-定義連線參數] 下的 Windows 設定，輸入您建立的資料來源名稱。同時輸入資料庫名稱、結構描述名稱、使用者名稱和密碼。選擇確定。
12. 等待企業分析器創建所有的表，索引，等等，它需要存儲結果。此過程可能需要幾分鐘的時間。企業分析器會確認資料庫和工作區何時可供使用。
13. 再次瀏覽至應用程式選取器功能表，然後選擇「企業分析器」以啟動工具。
14. 「企業分析器」啟動視窗會顯示在新選取的工作區位置中。選擇確定。
15. 導覽至左窗格中的儲存庫，選取儲存庫名稱，然後選擇「將檔案/資料夾新增至您的工作區」。選取儲存應用程式程式碼的資料夾，將其新增至工作區。如果需要，您可以使用前面的 BankDemo 示例代碼。當企業分析器提示您驗證這些檔案時，請選擇驗證以啟動初始的「企業分析器」驗證報告。可能需要幾分鐘的時間才能完成，具體取決於您的應用程序的大小。
16. 展開工作區以查看已新增至工作區的檔案和資料夾。物件類型和環狀複雜度報告也會顯示在 [圖表檢視器] 窗格的頂部象限中。

您現在可以使用企業分析器來執行所有必要的工作。

後續會話

1. 使用您在 AppStream 2.0 的歡迎電子郵件訊息中收到的 AppStream 2.0 開始工作階段。URL
2. 使用您的電子郵件和永久密碼登錄。
3. 選取企業分析器堆疊。
4. 如果您使用此選項共用工作區檔案，請啟動Rclone以連接到 Amazon S3 支援的磁碟。
5. 啟動企業分析器來執行您的任務。

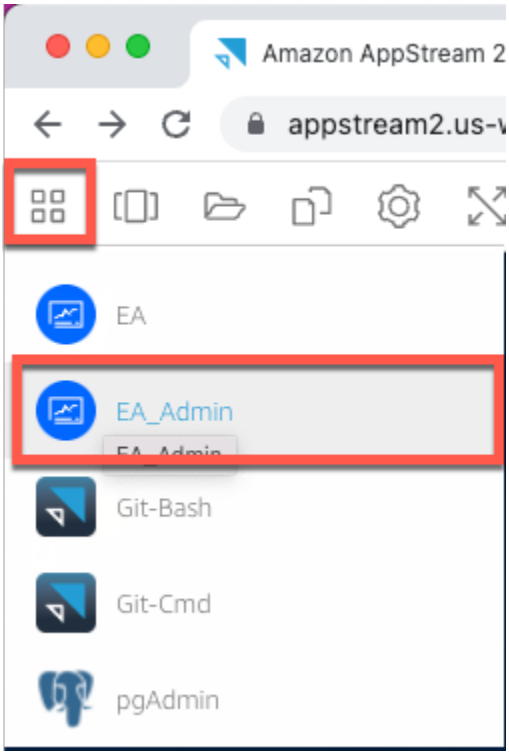
工作區連線疑難

當您嘗試重新連線到您的企業分析器工作區時，您可能會看到這樣的錯誤：

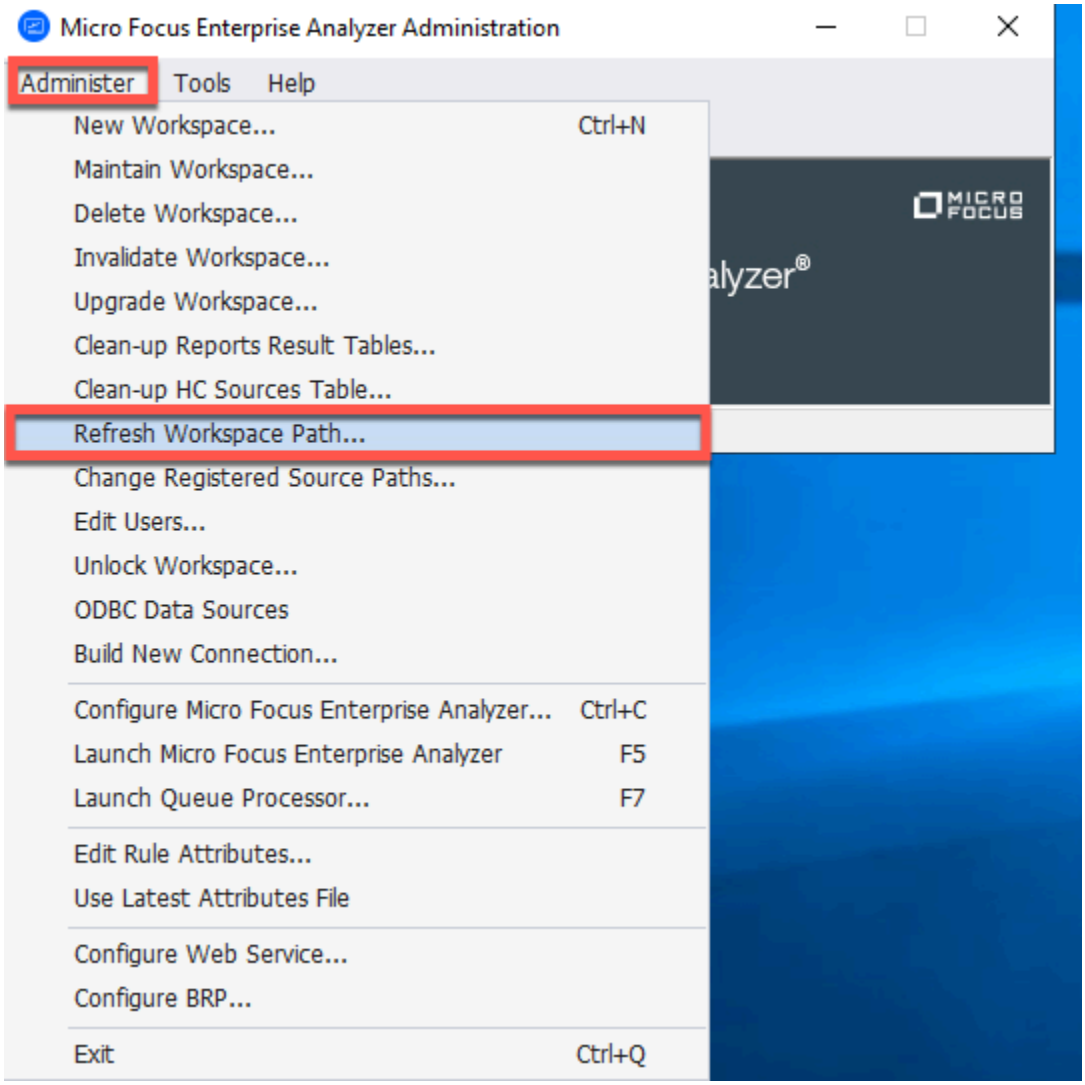
```
Cannot access the workspace directory D:\PhotonUser\My Files\Home Folder\EA_BankDemo.
 The workspace has been created on a non-shared disk of the EC2AMAZ-E6LC33H computer.
 Would you like to correct the workspace directory location?
```

若要解決此問題，請選擇 [確定] 清除訊息，然後完成下列步驟。

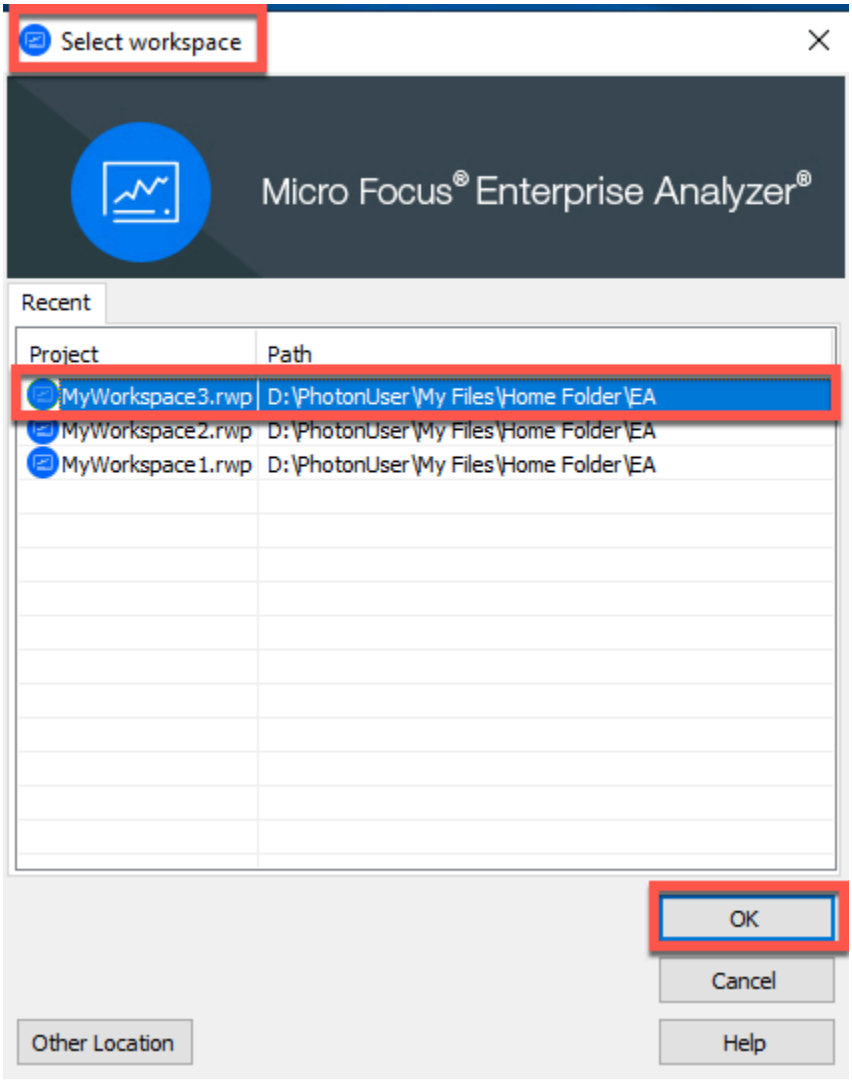
1. 在 AppStream 2.0 中，選擇工具列上的啟動應用程式圖示，然後選擇 EA_Admin 以啟動 Micro Focus 企業分析器管理工具。



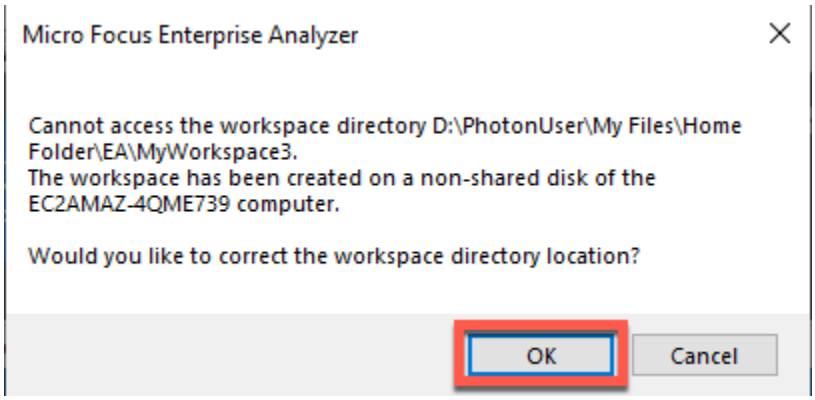
2. 從管理功能表中選擇重新整理工作區路徑... 。



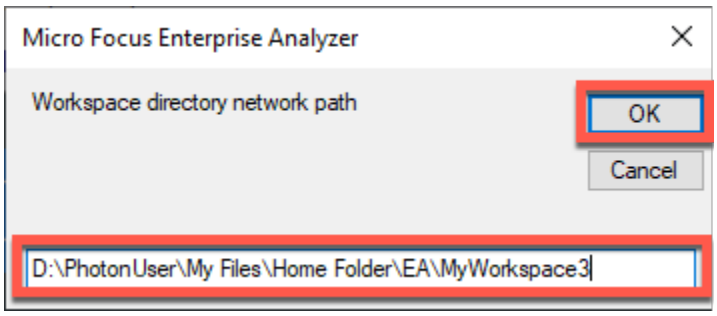
3. 在 [選取工作區] 底下，選擇您想要的工作區，然後選擇 [確定]。



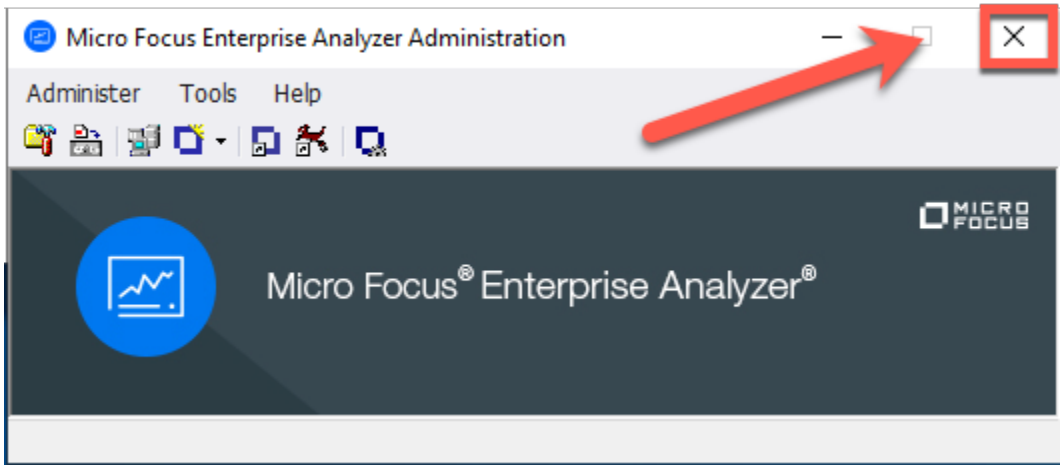
4. 選擇 「確定」 以確認錯誤訊息。



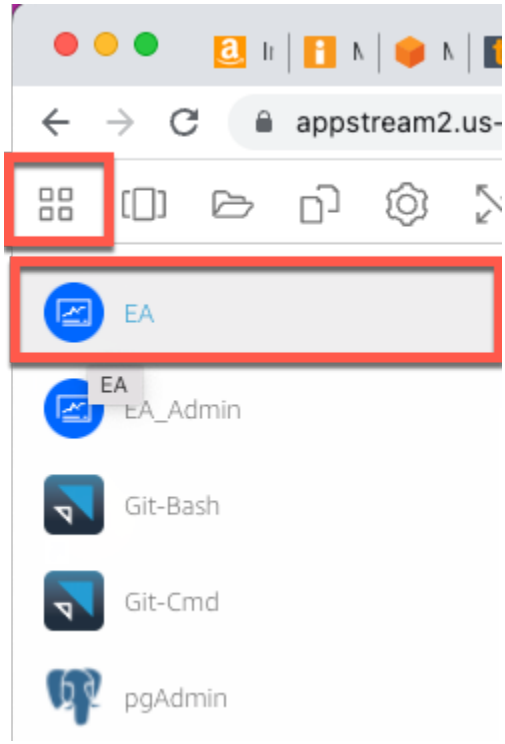
5. 在「工作區目錄網路路徑」下，輸入工作區的正確路徑，例如`D:\PhotonUser\My Files\Home Folder\EA\MyWorkspace3`。



6. 關閉 Micro Focus 企業分析器管理工具。



7. 在 AppStream 2.0 中，選擇工具列上的啟動應用程式圖示，然後選擇 EA 啟動 Micro Focus 企業分析器。



8. 重複步驟 3-5。

Micro Focus 企業分析器現在應該與現有的工作區打開。

清除資源

如果您不再需要為此教學課程建立的資源，請刪除這些資源，以免產生其他費用。請完成下列步驟：

- 使用 EA_Admin 工具刪除工作區。
- 刪除您為本教學課程建立的 S3 儲存貯體。如需詳細資訊，請參閱 Amazon S3 使用者指南中的刪除儲存貯體。
- 刪除您為此教學課程建立的資料庫。如需詳細資訊，請參閱刪除資料庫執行個體。

教學課程：在 AppStream 2.0 上設定微焦點企業開發人員

本教學課程說明如何為一或多個大型主機應用程式設定 Micro Focus 企業開發人員，以便使用企業開發人員功能來維護、編譯和測試這些應用程式。此設定是以 AWS 大型主機現代化與客戶共用的 AppStream 2.0 Windows 映像，以及如中所述建立 AppStream 2.0 叢集和堆疊的基礎。教學課程：設定 AppStream 2.0 以搭配 Micro Focus 企業分析儀和微焦點企業開發人員使用

> **Important**
>
> 本教學課程中的步驟假設您使用可下載的 AWS CloudFormation 範本 cfn-m2-appstream-fleet-ea-ed .yaml 設定 AppStream 2.0。如需詳細資訊，請參閱教學課程：設定 AppStream 2.0 以搭配 Micro Focus 企業分析儀和微焦點企業開發人員使用。
> 當企業開發人員叢集和堆疊啟動並執行時，您必須執行此設定的步驟。

如需企業開發人員第 7 版功能和交付項目的完整說明，請參閱 Micro Focus 網站上的up-to-date 線上文件 (第 7.0 版)。

圖片內容

除了企業開發人員本身，圖像包含包含倫巴（TN3270 模擬器）的圖像。它還包含以下工具和庫。

第三方工具

- Python

- 複製
- pgAdmin
- GIT 供應鏈管理
- 後驅動程序 SQL ODBC

中的圖書館 `C:\Users\Public`

- BankDemo 企業開發人員的源代碼和項目定義：`m2-bankdemo-template.zip`。
- MFA大型主機的安裝套件：。`mfa.zip`如需詳細資訊，請參閱 Micro Focus 企業開發人員文件中的大型主機存取概觀。
- Rclone 的命令和配置文件（在教程中使用它們的說明）：`m2-rclone.cmd`和`m2-rclone.conf`。

如果您需要存取尚未載入到 CodeCommit 儲存庫中但可在 Amazon S3 儲存貯體中使用的原始程式碼 (例如，將原始程式碼初始載入 git)，請依照中所述的程序建立虛擬 Windows 磁碟教學課程：在 AppStream 2.0 上設定企業分析器。

主題

- 必要條件
- 步驟 1：由個別企業開發者使用者進行設定
- 步驟 2：在 Windows 上創建基於亞馬遜 S3 的虛擬文件夾（可選）
- 步驟 3：克隆存儲庫
- 後續會話
- 清除資源

必要條件

- 一個或多個 CodeCommit 存儲庫加載了要維護的應用程序的源代碼。儲存庫設定應符合上述 CI/CD 管線的需求，以便結合兩種工具來建立協同效應。
- 每個使用者都必須具有由帳戶管理員根據的驗證和 CodeCommit 存取控制中的資訊所定義的儲存庫的認證AWS CodeCommit。這些證明資料的結構會在「認證和存取控制」中檢閱，IAM授權的完整參考資料請參考CodeCommit 權限參考資料：管理員可以為具有每個儲存區域角色特有之證明資料的不同角色定義不同的IAM原則，並將使用者的授權限制在指定儲存區域上必須完成的特定工作集。AWS CodeCommit CodeCommit 因此，對於 CodeCommit 存儲庫的每個維護者，帳戶

管理員將生成一個主要用戶，並通過選擇適當的訪問IAM策略授予該用戶訪問所需存儲庫的權限。CodeCommit

步驟 1：由個別企業開發者使用者進行設定

1. 取得您的IAM憑證：
   1. Connect 到 AWS 主控台的位置https://console.aws.amazon.com/iam/。
   2. 請遵循使用者指南中針對使用 Git 認證的使用HTTPS AWS CodeCommit 者的安裝步驟 3 中所述的程序進行。
   3. 複製為您IAM產生的 CodeCommit特定登入認證，方法是顯示、複製這些資訊，然後貼到本機電腦上的安全檔案中，或選擇 [下載認證] 將此資訊下載為. CSV文件。您需要此資訊才能連線到 CodeCommit。
2. 根據歡迎電子郵件中收到的 URL 以 AppStream 2.0 開始工作階段。使用您的電子郵件作為用戶名並創建密碼。
3. 選取您的企業開發人員堆疊。
4. 在功能表頁面上，選擇 [桌面] 以到達叢集串流處理的 Windows 桌面平台。

步驟 2：在 Windows 上創建基於亞馬遜 S3 的虛擬文件夾（可選）

如果需要 Rclone (請參閱上文)，請在 Windows 上建立以 Amazon S3 為基礎的虛擬資料夾：(如果所有應用程式成品僅來自存取，則為選用)。 CodeCommit

> **Note**
>
> 如果您已在 AWS 大型主機現代化預覽期間使用 Rclone，則必須更新`m2-rclone.cmd`至中的較新版本。`C:\Users\Public`

1. `C:\Users\PhotonUser\My Files\Home Folder`使用 [檔`m2-rclone.cmd`案總管] 將中提供的`m2-rclone.conf`和檔案複製`C:\Users\Public`到您的個人資料夾。
2. 使用`m2-rclone.conf`您的 AWS 訪問密鑰和相應的密鑰以及您的 AWS 區域.

```
[m2-s3]
type = s3
provider = AWS
```

```
access_key_id = YOUR-ACCESS-KEY
secret_access_key = YOUR-SECRET-KEY
region = YOUR-REGION
acl = private
server_side_encryption = AES256
```

3. 在 m2-rclone.cmd 中，進行下列變更：

   - 更改your-s3-bucket為您的 Amazon S3 存儲桶名稱。例如：m2-s3-mybucket。
   - 變更your-s3-folder-key為您的 Amazon S3 儲存貯體金鑰。例如：myProject。
   - 變更your-local-folder-path至您要從包含這些檔案的 Amazon S3 儲存貯體同步應用程式檔案的目錄路徑。例如：D:\PhotonUser\My Files\Home Folder\m2-new。此同步目錄必須是主資料夾的子目錄， AppStream 2.0 才能在作業階段開始和結束時正確備份和還原它。

```
:loop
timeout /T 10
"C:\Program Files\rclone\rclone.exe" sync m2-s3:your-s3-bucket/your-s3-folder-
key "D:\PhotonUser\My Files\Home Folder\your-local-folder-path" --config "D:
\PhotonUser\My Files\Home Folder\m2-rclone.conf"
goto :loop
```

4. 打開一個 Windows 命令提示符，如C:\Users\PhotonUser\My Files\Home Folder果需要，請點擊並運行m2-rclone.cmd。此命令指令碼會執行連續迴圈，每 10 秒將 Amazon S3 儲存貯體和金鑰同步到本機資料夾。您可以根據需要調整逾時。您應該會看到位於 Windows 檔案總管中 Amazon S3 儲存貯體中的應用程式原始程式碼。

若要將新檔案新增至您正在處理的檔案集或更新現有檔案，請將檔案上傳到 Amazon S3 儲存貯體，這些檔案會在定義的下一次迭代時同步到您的目錄m2-rclone.cmd。同樣地，如果您想要刪除某些檔案，請從 Amazon S3 儲存貯體中刪除這些檔案。下一個同步操作將從您的本地目錄中刪除它們。

步驟 3：克隆存儲庫

1. 瀏覽至瀏覽器視窗左上角的應用程式選取器功能表，然後選取企業開發人員。
2. 選擇 C:\Users\PhotonUser\My Files\Home Folder (又名D: \PhotonUser\My Files\Home Folder) 做為工作區的位置，完成企業開發人員在 [首頁] 資料夾中所需的工作區建立。
3. 在企業開發人員中，通過轉到項目資源管理器克隆您的 CodeCommit 存儲庫，右鍵單擊並選擇導入，導入...，Git，從 Git 克隆項目URI。然後，輸入您的 CodeCommit特定登錄憑據並完成 Eclipse 對話框以導入代碼。

中的 CodeCommit git 存儲庫現在克隆到本地工作區中。

您的企業開發人員工作區現在已準備就緒，可在應用程式上開始維護工作。特別是，您可以使用與企業開發人員整合的 Microfocus 企業伺服器 (ES) 的本機執行個體，以互動方式偵錯和執行應用程式，以便在本機驗證您的變更。

> **Note**
>
> 本機企業開發人員環境 (包括本機企業伺服器執行個體) 在 Windows 下執行，而 AWS 大型主機現代化則在 Linux 下執行。建議您在為此目標認可新應用程式 CodeCommit 並重建新應用程式之後，以及在將新應用程式推出至生產環境之前，在 AWS 大型主機現代化提供的 Linux 環境中執行互補測試。

後續會話

當您選取 AppStream 2.0 管理下的資料夾 (例如用於複製存 CodeCommit 放庫的主資料夾) 時，會在工作階段間透明地儲存和還原該資料夾。下次需要使用應用程式時，請完成下列步驟：

1. 根據歡迎電子郵件中收到的 URL 以 AppStream 2.0 開始工作階段。
2. 使用您的電子郵件和永久密碼登錄。
3. 選取企業開發人員堆疊。
4. 使用此選項`Rclone`來共用工作區檔案時，啟動以連線 (請參閱上文) 到 Amazon S3 支援的磁碟。
5. 啟動企業開發人員來完成您的工作。

清除資源

如果您不再需要為此教學課程建立的資源，請刪除這些資源，以免繼續支付這些資源的費用。請完成下列步驟：

- 刪除您為此教學課程建立的 CodeCommit 儲存庫。如需詳細資訊，請參閱《AWS CodeCommit 使用指南》中的「刪除 CodeCommit 存放庫」。
- 刪除您為此教學課程建立的資料庫。如需詳細資訊，請參閱刪除資料庫執行個體。

AWS 大型主機現代化中可用的批次公用程式

大型主機應用程式通常使用批次公用程式來執行特定功能，例如排序資料、使用傳輸檔案FTP、將資料載入資料到資料庫DB2，以及從資料庫卸載資料等等。

將應用程式移轉至 AWS 大型主機現代化時，您需要功能上等同的替換公用程式，以執行與您在大型主機上使用的工作相同的工作。其中一些公用程式可能已經作為 AWS 大型主機現代化執行階段引擎的一部分提供，但我們提供下列替代公用程式：

- M2 SFTP-啟用使用SFTP協議的安全文件傳輸。
- M2 WAIT-等待指定的時間量繼續在批處理作業的下一個步驟之前。
- TXT2PDF-將文本文件轉換為PDF格式。
- M2 DFUTIL-提供與大型主機ADRDSSU實用程序提供的支持類似的數據集備份，還原，刪除和複製功能。
- M2 RUNCMD-可讓您直接從執行微焦點指令、指令碼和系統呼叫JCL。

我們根據客戶意見反應開發這些批次公用程式，並將其設計為提供與大型主機公用程式相同的功能。我們的目標是讓您從大型主機到大型主機現代化的過渡盡可能 AWS 順利。

主題

- 二進位位置
- M2 SFTP 批次工具
- M2 WAIT 批次工具
- TXT2PDF批次程式
- M2 DFUTIL 批次工具
- M2 RUNCMD 批次工具

二進位位置

這些公用程式已預先安裝在微焦點企業開發人員 (ED) 和微焦點企業伺服器 (ES) 產品上。您可以在以下位置找到它們的 ED 和 ES 的所有變體：

- Linux：`/opt/aws/m2/microfocus/utilities/64bit`
- 視窗 (32 位元): `C:\AWS\M2\MicroFocus\Utilities\32bit`
- 視窗 (64 位元): `C:\AWS\M2\MicroFocus\Utilities\64bit`

M2 SFTP 批次工具

M2 SFTP 是一個JCL實用程序，旨在使用安全文件傳輸協議（SFTP）系統之間執行安全文件傳輸。該程序使用 Putty SFTP 客戶端psftp，執行實際的文件傳輸。該程序的工作方式類似於大型主機實用程FTP序，並使用用戶和密碼身份驗證。

> **Note**
>
> 不支援公開金鑰驗證。

若要將大型主機轉換為使FTPJCLs用SFTP，請變更PGM=FTP為。PGM=M2SFTP

主題

- 支援平台
- 安裝相依項目
- SFTP為受管理的 AWS 大型主機現代化設定 M2
- 在 Amazon 上SFTP為 AWS 大型主機現代化運行時配置 M2EC2（包括 2.0） AppStream
- 樣品 JCLs
- 膩子 SFTP (PSFTP) 用戶端命令參考
- 後續步驟

支援平台

您可以在以下任何平台SFTP上使用 M2：

- AWS 大型主機現代化微焦點管理
- 微焦點運行時（在 Amazon 上EC2）
- 微焦點企業開發商（ED）和微焦點企業服務器（ES）產品的所有變體。

安裝相依項目

若要在視窗上安裝膩子SFTP用戶端

- 下載 P u TTY SFTP 客戶端並安裝它。

若要在 Linux 上安裝膩子SFTP用戶端：

- 執行下列命令以安裝 Putty SFTP 用戶端：

```
sudo yum -y install putty
```

SFTP為受管理的 AWS 大型主機現代化設定 M2

如果您移轉的應用程式是在受管理的 AWS 大型主機現代化上執行，您將需要設定 M2SFTP，如下所示。

- 設定適當的 Micro Focus 企業伺服器環境變數MFFTP。這裡有幾個例子：
  - `MFFTP_TEMP_DIR`
  - `MFFTP_SENDEOL`
  - `MFFTP_TIME`
  - `MFFTP_ABEND`

  您可以根據需要設置為少或盡可能多的這些變量。您可以在JCL使用ENVAR DD語句中設置它們。如需這些變數的詳細資訊，請參閱 Micro Focus 文件中的MFFTP控制變數。

若要測試您的組態，請參閱樣品 JCLs。

在 Amazon 上SFTP為 AWS 大型主機現代化運行時配置 M2EC2（包括 2.0）AppStream

如果您遷移的應用程式在 Amazon 上的 AWS 大型主機現代化執行階段上執行EC2，請按如下方式設定 M2 SFTP。

1. 變更 Micro Focus JES 程式路徑以包含批次公用程式的二進位位置。如果您需要指定多個路徑，請使用冒號 (`:`) 來分隔 Linux 上的路徑，在 Windows 上使用分號 (`;`) 來分隔路徑。
   - Linux：`/opt/aws/m2/microfocus/utilities/64bit`
   - 視窗 (32 位元): `C:\AWS\M2\MicroFocus\Utilities\32bit`
   - 視窗 (64 位元): `C:\AWS\M2\MicroFocus\Utilities\64bit`

2. 設定適當的 Micro Focus 企業伺服器環境變數MFFTP。這裡有幾個例子：

   - MFFTP_TEMP_DIR
   - MFFTP_SENDEOL
   - MFFTP_TIME
   - MFFTP_ABEND

   您可以根據需要設置為少或盡可能多的這些變量。您可以在JCL使用ENVAR DD語句中設置它們。如需這些變數的詳細資訊，請參閱 Micro Focus 文件中的MFFTP控制變數。

若要測試您的組態，請參閱樣品 JCLs。

樣品 JCLs

若要測試安裝，您可以使用下列其中一個範例JCL檔案。

平方SFTP1米

這JCL顯示了如何調用 M2 SFTP 將文件發送到遠程SFTP服務器。請注意ENVVAR DD陳述式中設定的環境變數。

```
//M2SFTP1 JOB 'M2SFTP1',CLASS=A,MSGCLASS=X,TIME=1440
//*
//* Copyright Amazon.com, Inc. or its affiliates.*
//* All Rights Reserved.*
//*
//*-------------------------------------------------------------------**
//* Sample SFTP JCL step to send a file to SFTP server*
//*-------------------------------------------------------------------**
//*
//STEP01 EXEC PGM=M2SFTP,
//            PARM='127.0.0.1 (EXIT=99 TIMEOUT 300'
//*
//SYSFTPD  DD  *
RECFM FB
LRECL 80
SBSENDEOL CRLF
MBSENDEOL CRLF
TRAILINGBLANKS FALSE
/*
//NETRC    DD  *
```

```
machine 127.0.0.1 login sftpuser password sftppass
/*
//SYSPRINT DD  SYSOUT=*
//OUTPUT   DD  SYSOUT=*
//STDOUT   DD  SYSOUT=*
//INPUT    DD  *
type a
locsite notrailingblanks
cd files
put 'AWS.M2.TXT2PDF1.PDF' AWS.M2.TXT2PDF1.pdf
put 'AWS.M2.CARDDEMO.CARDDATA.PS' AWS.M2.CARDDEMO.CARDDATA.PS1.txt
quit
/*
//ENVVAR   DD *
MFFTP_VERBOSE_OUTPUT=ON
MFFTP_KEEP=N
/*
//*
//
```

平方SFTP2米

這JCL顯示了如何調用 M2 SFTP 從遠程SFTP服務器接收文件。請注意ENVVAR DD陳述式中設定的環境變數。

```
//M2SFTP2 JOB 'M2SFTP2',CLASS=A,MSGCLASS=X,TIME=1440
//*
//* Copyright Amazon.com, Inc. or its affiliates.*
//* All Rights Reserved.*
//*
//*-------------------------------------------------------------------**
//* Sample SFTP JCL step to receive a file from SFTP server*
//*-------------------------------------------------------------------**
//*
//STEP01 EXEC PGM=M2SFTP
//*
//SYSPRINT DD  SYSOUT=*
//OUTPUT   DD  SYSOUT=*
//STDOUT   DD  SYSOUT=*
//INPUT    DD  *
open 127.0.0.1
sftpuser
sftppass
```

```
cd files
locsite recfm=fb lrecl=150
get AWS.M2.CARDDEMO.CARDDATA.PS.txt +
'AWS.M2.CARDDEMO.CARDDATA.PS2' (replace
quit
/*
//ENVVAR   DD *
MFFTP_VERBOSE_OUTPUT=ON
MFFTP_KEEP=N
/*
//*
//
```

> **Note**
>
> 我們強烈建議您將FTP認證儲存在NETRC檔案中，並限制只有授權使用者存取。

膩子 SFTP (PSFTP) 用戶端命令參考

用PSFTP戶端不支援所有FTP指令。下列清單顯示所有支PSFTP援的指令。

Command	描述
!	執行本機命令
再見	完成SFTP工作階段
光碟	變更遠端工作目錄
chmod	變更檔案權限和模式
關閉	完成您的SFTP工作階段，但不要結束 PSFTP
德爾	刪除遠端伺服器上的檔案
dir	列出遠端檔案
exit	完成SFTP工作階段
get	將檔案從伺服器下載到本機電腦

Command	描述
help	給予幫助
液晶	變更本機工作目錄
流行的	列印本機工作目錄
ls	列出遠端檔案
mget	一次下載多個文件
mkdir	在遠端伺服器上建立目錄
mput	一次上傳多個文件
MV	移動或重新命名遠端伺服器上的檔案
開啟	Connect 至主機
put	將文件從本地計算機上傳到服務器
PWD	列印您的遠端工作目錄
結束	完成SFTP工作階段
重獲	繼續下載檔案
仁	移動或重新命名遠端伺服器上的檔案
重新放入	繼續上傳檔案
RM	刪除遠端伺服器上的檔案
RMDIR	移除遠端伺服器上的目錄

後續步驟

若要使用將檔案上傳和下載到 Amazon 簡單儲存服務SFTP，您可以使用 M2 SFTP 結合使用 AWS Transfer Family，如下列部落格文章所述。

- 使用 AWS SFTP邏輯目錄建立簡單的資料散發服務
- 啟用密碼驗證以便 AWS Transfer for SFTP 使用 AWS Secrets Manager

M2 WAIT 批次工具

M2 WAIT 是一個大型主機實用程序，使您能夠通過指定以秒，分鐘或小時為單位的持續時間來引入JCL腳本中的等待期。您可以通JCL過將要等待的時間作為輸入參數傳遞來WAIT直接調用 M2。在內部，M2 WAIT 程序調用 Micro Focus 提供C$SLEEP的模塊等待指定的時間。

> **Note**
>
> 您可以使用 Micro Focus 別名來取代JCL指令碼中所擁有的項目。如需詳細資訊，請參閱 Micro Focus 文件中的JES別名。

主題

- 支援平台
- WAIT為受管理的 AWS 大型主機現代化設定 M2
- 在 Amazon 上WAIT為 AWS 大型主機現代化運行時配置 M2EC2（包括 2.0） AppStream
- 樣品 JCL

支援平台

您可以在以下任何平台WAIT上使用 M2：

- AWS 大型主機現代化微焦點管理
- 微焦點運行時（在 Amazon 上EC2）
- 微焦點企業開發商（ED）和微焦點企業服務器（ES）產品的所有變體。

WAIT為受管理的 AWS 大型主機現代化設定 M2

如果您移轉的應用程式是在受管理的 AWS 大型主機現代化上執行，您將需要設定 M2WAIT，如下所示。

- 通過傳遞輸入參數JCL來使用程序 M2WAIT，如圖所示樣品 JCL。

在 Amazon 上WAIT為 AWS 大型主機現代化運行時配置 M2EC2（包括 2.0）AppStream

如果您遷移的應用程式在 Amazon 上的 AWS 大型主機現代化執行階段上執行EC2，請按如下方式設定 M2 WAIT。

1. 變更 Micro Focus JES 程式路徑以包含批次公用程式的二進位位置。如果您需要指定多個路徑，請使用冒號 (`:`) 來分隔 Linux 上的路徑，在 Windows 上使用分號 (`;`) 來分隔路徑。
   - Linux：`/opt/aws/m2/microfocus/utilities/64bit`
   - 視窗 (32 位元): `C:\AWS\M2\MicroFocus\Utilities\32bit`
   - 視窗 (64 位元): `C:\AWS\M2\MicroFocus\Utilities\64bit`
2. 通過如圖所示傳遞輸入參數，WAIT在您JCL的使用程序 M2 樣品 JCL。

樣品 JCL

要測試安裝，您可以使用該`M2WAIT1.jcl`程序。

此範例JCL顯示如何呼叫 M2 WAIT 並傳遞數個不同的持續時間。

```
//M2WAIT1 JOB 'M2WAIT',CLASS=A,MSGCLASS=X,TIME=1440
//*
//* Copyright Amazon.com, Inc. or its affiliates.*
//* All Rights Reserved.*
//*
//*-------------------------------------------------------------------**
//* Wait for 12 Seconds*
//*-------------------------------------------------------------------**
//*
//STEP01 EXEC PGM=M2WAIT,PARM='S012'
//SYSOUT DD SYSOUT=*
//*
//*-------------------------------------------------------------------**
//* Wait for 0 Seconds (defaulted to 10 Seconds)*
//*-------------------------------------------------------------------**
//*
//STEP02 EXEC PGM=M2WAIT,PARM='S000'
//SYSOUT DD SYSOUT=*
//*
//*-------------------------------------------------------------------**
```

```
//* Wait for 1 Minute*
//*-------------------------------------------------------------------**
//*
//STEP03 EXEC PGM=M2WAIT,PARM='M001'
//SYSOUT DD SYSOUT=*
//*
//
```

TXT2PDF批次程式

TXT2PDF是一個通常用於將文本文件轉換為文件的大型主機實用程序。PDF此實用程序使用相同的源代碼TXT2PDF（z/OS 免費軟件）。 AWS 我們將其修改為在大型主機現代化 Micro Focus 運行時環境下運行。

主題

- 支援平台
- 設定TXT2PDF受管理的 AWS 大型主機現代化
- 在 Amazon 上TXT2PDF設定 AWS 大型主機現代化執行階段 EC2 (包括 2.0) AppStream
- 樣品 JCL
- 修改
- 參考

支援平台

您可以TXT2PDF在以下任何平台上使用：

- AWS 大型主機現代化微焦點管理
- 微焦點運行時（在 Amazon 上EC2）
- 微焦點企業開發商（ED）和微焦點企業服務器（ES）產品的所有變體。

設定TXT2PDF受管理的 AWS 大型主機現代化

如果您移轉的應用程式是在受管理的 AWS 大型主機現代化上執行，請依下列方式進行設定 TXT2PDF

- 建立名為的REXXEXEC程式庫`AWS.M2.REXX.EXEC`。下載這些REXX模塊並將其複製到庫中。
  - `TXT2PDF.rex`-TXT2PDF z/OS 免費軟件（已修改）

- TXT2PDFD.rex-TXT2PDF z/OS 免費軟體 (未修改)
- TXT2PDFX.rex-TXT2PDF z/OS 免費軟件（已修改）
- M2GETOS.rex-檢查操作系統類型（視窗或 Linux）

若要測試您的組態，請參閱樣品 JCL。

在 Amazon 上TXT2PDF設定 AWS 大型主機現代化執行階段 EC2 (包括 2.0) AppStream

如果您遷移的應用程式在 Amazon 的 AWS 大型主機現代化執行階段上執行EC2，請按照以下方式進行設定TXT2PDF。

1. MFREXX_CHARSET將 Micro Focus 環境變數設定為適當的值，例如「A" 表示資ASCII料。

   > **Important**
   >
   > 輸入錯誤的值可能會導致資料轉換問題 (從EBCDIC到ASCII)，導致PDF無法讀取或無法運作。我們建議設MFREXX_CHARSET置匹配MF_CHARSET。

2. 變更 Micro Focus JES 程式路徑以包含批次公用程式的二進位位置。如果您需要指定多個路徑，請使用冒號 (:) 來分隔 Linux 上的路徑，在 Windows 上使用分號 (;) 來分隔路徑。

   - Linux：/opt/aws/m2/microfocus/utilities/64bit
   - 視窗 (32 位元): C:\AWS\M2\MicroFocus\Utilities\32bit
   - 視窗 (64 位元): C:\AWS\M2\MicroFocus\Utilities\64bit

3. 建立名為的REXXEXEC程式庫AWS.M2.REXX.EXEC`。下載這些REXX模塊並將其複製到庫中。

   - TXT2PDF.rex-TXT2PDF z/OS 免費軟件（已修改）
   - TXT2PDFD.rex-TXT2PDF z/OS 免費軟體 (未修改)
   - TXT2PDFX.rex-TXT2PDF z/OS 免費軟件（已修改）
   - M2GETOS.rex-檢查操作系統類型（視窗或 Linux）

若要測試您的組態，請參閱樣品 JCL。

樣品 JCL

若要測試安裝，您可以使用下列其中一個範例JCL檔案。

TXT2PDF1.jcl

此範例JCL檔案使用 DD 名稱進行TXT2PDF轉換。

```
//TXT2PDF1 JOB 'TXT2PDF1',CLASS=A,MSGCLASS=X,TIME=1440
//*
//* Copyright Amazon.com, Inc. or its affiliates.*
//* All Rights Reserved.*
//*
//*--------------------------------------------------------------------**
//* PRE DELETE*
//*--------------------------------------------------------------------**
//*
//PREDEL  EXEC PGM=IEFBR14
//*
//DD01     DD DSN=AWS.M2.TXT2PDF1.PDF.VB,
//            DISP=(MOD,DELETE,DELETE)
//*
//DD02     DD DSN=AWS.M2.TXT2PDF1.PDF,
//            DISP=(MOD,DELETE,DELETE)
//*
//*--------------------------------------------------------------------**
//* CALL TXT2PDF TO CONVERT FROM TEXT TO PDF (VB)*
//*--------------------------------------------------------------------**
//*
//STEP01 EXEC PGM=IKJEFT1B
//*
//SYSEXEC  DD DISP=SHR,DSN=AWS.M2.REXX.EXEC
//*
//INDD     DD *
1THIS IS THE FIRST LINE ON THE PAGE 1
0THIS IS THE THIRD LINE ON THE PAGE 1
-THIS IS THE   6TH LINE ON THE PAGE 1
THIS IS THE   7TH LINE ON THE PAGE 1
+_____________________________________ - OVERSTRIKE 7TH LINE
1THIS IS THE FIRST LINE ON THE PAGE 2
0THIS IS THE THIRD LINE ON THE PAGE 2
-THIS IS THE   6TH LINE ON THE PAGE 2
THIS IS THE   7TH LINE ON THE PAGE 2
+_____________________________________ - OVERSTRIKE 7TH LINE
/*
//*
//OUTDD    DD DSN=AWS.M2.TXT2PDF1.PDF.VB,
//            DISP=(NEW,CATLG,DELETE),
```

```
//            DCB=(LRECL=256,DSORG=PS,RECFM=VB,BLKSIZE=0)
//*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DDNAME=SYSIN
//*
//SYSIN    DD *
%TXT2PDF BROWSE Y IN DD:INDD +
OUT DD:OUTDD +
CC YES
/*
//*
//*--------------------------------------------------------------------**
//* CONVERT PDF (VB) TO PDF (LSEQ - BYTE STREAM)*
//*--------------------------------------------------------------------**
//*
//STEP02 EXEC PGM=VB2LSEQ
//*
//INFILE   DD DSN=AWS.M2.TXT2PDF1.PDF.VB,DISP=SHR
//*
//OUTFILE  DD DSN=AWS.M2.TXT2PDF1.PDF,
//            DISP=(NEW,CATLG,DELETE),
//            DCB=(LRECL=256,DSORG=PS,RECFM=LSEQ,BLKSIZE=0)
//*
//SYSOUT   DD SYSOUT=*
//*
//
```

TXT2PDF2.jcl

此範例JCL使用DSN名稱進行TXT2PDF轉換。

```
//TXT2PDF2 JOB 'TXT2PDF2',CLASS=A,MSGCLASS=X,TIME=1440
//*
//* Copyright Amazon.com, Inc. or its affiliates.*
//* All Rights Reserved.*
//*
//*--------------------------------------------------------------------**
//* PRE DELETE*
//*--------------------------------------------------------------------**
//*
//PREDEL  EXEC PGM=IEFBR14
//*
//DD01     DD DSN=AWS.M2.TXT2PDF2.PDF.VB,
//            DISP=(MOD,DELETE,DELETE)
```

```
//*
//DD02     DD DSN=AWS.M2.TXT2PDF2.PDF,
//            DISP=(MOD,DELETE,DELETE)
//*
//*--------------------------------------------------------------------**
//* CALL TXT2PDF TO CONVERT FROM TEXT TO PDF (VB)*
//*--------------------------------------------------------------------**
//*
//STEP01 EXEC PGM=IKJEFT1B
//*
//SYSEXEC  DD DISP=SHR,DSN=AWS.M2.REXX.EXEC
//*
//INDD     DD *
1THIS IS THE FIRST LINE ON THE PAGE 1
0THIS IS THE THIRD LINE ON THE PAGE 1
-THIS IS THE   6TH LINE ON THE PAGE 1
THIS IS THE   7TH LINE ON THE PAGE 1
+_____________________________________ - OVERSTRIKE 7TH LINE
1THIS IS THE FIRST LINE ON THE PAGE 2
0THIS IS THE THIRD LINE ON THE PAGE 2
-THIS IS THE   6TH LINE ON THE PAGE 2
THIS IS THE   7TH LINE ON THE PAGE 2
+_____________________________________ - OVERSTRIKE 7TH LINE
/*
//*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DDNAME=SYSIN
//*
//SYSIN    DD *
%TXT2PDF BROWSE Y IN DD:INDD +
OUT 'AWS.M2.TXT2PDF2.PDF.VB' +
CC YES
/*
//*
//*--------------------------------------------------------------------**
//* CONVERT PDF (VB) TO PDF (LSEQ - BYTE STREAM)*
//*--------------------------------------------------------------------**
//*
//STEP02 EXEC PGM=VB2LSEQ
//*
//INFILE   DD DSN=AWS.M2.TXT2PDF2.PDF.VB,DISP=SHR
//*
//OUTFILE  DD DSN=AWS.M2.TXT2PDF2.PDF,
//            DISP=(NEW,CATLG,DELETE),
```

```
//            DCB=(LRECL=256,DSORG=PS,RECFM=LSEQ,BLKSIZE=0)
//*
//SYSOUT   DD SYSOUT=*
//*
//
```

修改

為了讓TXT2PDF程式在 AWS 大型主機現代化 Micro Focus 執行階段環境中執行，我們進行了下列變更：

- 變更原始程式碼以確保與 Micro Focus REXX 執行階段的相容性
- 更改以確保程序可以在 Windows 和 Linux 操作系統上運行
- 支援EBCDIC和ASCII執行階段的修改

參考

TXT2PDF引用和源代碼：

- 文本到轉PDF換器
- z/OS 免費軟體 TCP /IP 和郵件工具
- TXT2PDF使用者參考指南

M2 DFUTIL 批次工具

M2 DFUTIL 是一個JCL實用程序，提供數據集的備份，還原，刪除和複製功能，類似於大型主機ADRDSSU實用程序提供的支持。該程序保留了許多SYSIN參數ADRDSSU，從而簡化了遷移到此新公用程序的過程。

主題

- 支援平台
- 平台需求
- 計劃的 future 支持
- 資產位置
- 在 Amazon 上設定 M2 DFUTIL 或 AWS 大型主機現代化執行階段 EC2 (包括 2.0) AppStream
- 一般語法

- 樣品 JCLs

支援平台

您可以在以下任何平台DFUTIL上使用 M2：

- 微焦點 ES 視窗 (64 位元和 32 位元)
- 微焦點 ES 在 Linux 上 (64 位元)

平台需求

M2 DFUTIL 取決於調用腳本來執行正則表達式測試。在視窗上，您必須安裝 Linux 版視窗服務 (WSL) 才能執行此指令碼。

計劃的 future 支持

目前無法從大型主機ADRDSSU公用程式使用，但在 future 範圍內的功能包括：

- M2 管理
- VSAM
- COPY支持文件名重命名
- RENAME支援 RESTORE
- 多重INCLUDE和 EXCLUDE
- BY 子句用於子選取DSORG,, CREDT EXPDT
- MWAIT重試排隊失敗的子句
- S3 儲存支援DUMP/RESTORE

資產位置

此公用程式的載入模組會`M2DFUTIL.so`在 Linux 和視窗`M2DFUTIL.dll`上呼叫。您可以在下列位置找到此載入模組：

- Linux：`/opt/aws/m2/microfocus/utilities/64bit`
- 視窗 (32 位元): `C:\AWS\M2\MicroFocus\Utilities\32bit`
- 視窗 (64 位元): `C:\AWS\M2\MicroFocus\Utilities\64bit`

用於正則表達式測試的腳本被調用compare.sh。您可以在下列位置找到此指令碼：

- Linux：/opt/aws/m2/microfocus/utilities/scripts
- 視窗 (32 位元): C:\AWS\M2\MicroFocus\Utilities\scripts

在 Amazon 上設定 M2 DFUTIL 或 AWS 大型主機現代化執行階段 EC2 (包括 2.0) AppStream

使用下列項目設定您的企業伺服器區域：

- 在 [ES 環境] 中添加以下變量
  - M2DFUTILS_BASE_LOC-DUMP 輸出的默認位置
  - M2DFUTILS_SCRIPTPATH-資產位置中記錄的**compare.sh**腳本位置
  - M2DFUTILS_VERBOSE-[VERBOSE或NORMAL]。這控制SYSPRINT 輸出中的細節級別
- 確認載入模組路徑已新增至JES\Configuration\JES Program Path設定
- 確認公用程式目錄中的指令碼具有執行權限。您可以在 Linux 環境中使用chmod + x <script name>命令新增執行權限

一般語法

DUMP

提供將檔案從目前編目位置複製到備份位置的功能。此位置目前必須是檔案系統。

處理

DUMP將執行以下操作：

1. 建立目標位置目錄。
2. 將目標位置目錄編目為PDS成員。
3. 透過處理INCLUDE參數來決定要包含的檔案。
4. 透過處理EXCLUDE參數來取消選取包含的檔案。
5. 確定是否要轉儲的文件。DELETED
6. 排隊要處理的文件。
7. 複製檔案。
8. 將複製的檔案目錄DCB資訊匯出至目標位置的副檔案，以協助 future RESTORE 作業。

語法

```
DUMP
TARGET ( TARGET LOCATION  )    -
INCLUDE ( DSN. )
[ EXCLUDE ( DSN ) ]
[ CANCEL | IGNORE ]
[ DELETE ]
```

必要參數

以下是所需的參數DUMP：

- SYSPRINT DD NAME-若要包含其他記錄資訊
- TARGET-目標位置。它可以是：
  - 傾印位置的完整路徑
  - 在 M2 中定義的位置創建的子目錄名稱 DFUTILS _ BASE _ LOC 變量
- INCLUDE-單一具名DSNAME或有效的大型主機搜尋字串 DSN
- EXCLUDE-單一具名DSNAME或有效的大型主機搜尋字串 DSN

選用的參數

- CANCEL-如果發生任何錯誤，請取消。將保留已處理的檔案
- （默認）IGNORE-忽略任何錯誤和過程，直到結束
- DELETE-如果沒有發生ENQ錯誤，則文件被刪除並取消編目

DELETE

提供批量刪除和取消編目文件的功能。不備份檔案。

處理

DELETE將執行以下操作：

1. 透過處理INCLUDE參數來決定要包含的檔案。
2. 透過處理EXCLUDE參數來取消選取包含的檔案。
3. 排隊要處理的文件。將處理方式設定為OLDDELETE、KEEP。

語法

```
DELETE
INCLUDE ( DSN )
[ EXCLUDE ( DSN ) ]
[ CANCEL | IGNORE ]
[ DELETE ]
```

必要參數

以下是所需的參數DELETE：

- SYSPRINT DD NAME-若要包含其他記錄資訊
- INCLUDE-單一具名DSNAME或有效的大型主機搜尋字串 DSN
- EXCLUDE-單一具名DSNAME或有效的大型主機搜尋字串 DSN

選用的參數

- CANCEL-如果發生任何錯誤，請取消。被處理的文件將被保留
- （默認）IGNORE-忽略任何錯誤和過程，直到結束

RESTORE

提供還原先前使用備份的檔案的功能DUMP。除非用於變更還原的檔案，否則RENAME檔案會還DSNAME原到原始的目錄位置。

處理

RESTORE將執行以下操作：

1. 驗證來源位置目錄。
2. 透過處理型錄匯出檔案來決定要包含的檔案。
3. 透過處理EXCLUDE參數來取消選取包含的檔案。
4. 排隊要處理的文件。
5. 不根據匯出資訊編目的目錄檔案。
6. 如果檔案已編目且匯出目錄資訊相同，則如果已設定REPLACE選項，則RESTORE會取代已編目的資料集。

語法

```
RESTORE
SOURCE ( TARGET LOCATION )
INCLUDE ( DSN )
[ EXCLUDE ( DSN ) ]
[ CANCEL | IGNORE ]
[ REPLACE]
```

必要參數

以下是所需的參數RESTORE：

- SYSPRINT DD NAME-若要包含其他記錄資訊
- SOURCE-源位置。它可以是：
  - 傾印位置的完整路徑
  - 在 M2 中定義的位置創建的子目錄名稱 DFUTILS _ BASE _ LOC 變量
- INCLUDE-單一具名DSNAME或有效的大型主機搜尋字串 DSN
- EXCLUDE-單一具名DSNAME或有效的大型主機搜尋字串 DSN

選用的參數

- CANCEL-如有任何錯誤，請取消。保留處理的檔案
- （默認）IGNORE-忽略任何錯誤和過程，直到結束
- REPLACE-如果要還原的檔案已編目且目錄記錄相同，則取代已編目的檔案

樣品 JCLs

DUMP工作

這項工作將建立名為TESTDUMP的子目錄。這是 M2 DFUTILS _ BASE _ LOC 變數指定的預設備份位置。它將為此備份創建一個名為的PDS庫M2DFUTILS.TESTDUMP。匯出的目錄資料會儲存在名為的備份目錄中的行順序檔案中CATDUMP.DAT。所有選取的檔案都會複製到此備份目錄。

```
//M2DFDMP JOB 'M2DFDMP',CLASS=A,MSGCLASS=X
//STEP001  EXEC PGM=M2DFUTIL
//SYSPRINT DD DSN=TESTDUMP.SYSPRINT,
```

```
//        DISP=(NEW,CATLG,DELETE),
//        DCB=(RECFM=LSEQ,LRECL=256)
//SYSIN    DD *
DUMP TARGET(TESTDUMP)               -
     INCLUDE(TEST.FB.FILE*.ABC)     -
 CANCEL
/*
//
```

DELETE工作

此工作將刪除目錄中符合INCLUDE參數的所有檔案。

```
/M2DFDEL JOB 'M2DFDEL',CLASS=A,MSGCLASS=X
//STEP001  EXEC PGM=M2DFUTIL
//SYSPRINT DD DSN=TESTDEL.SYSPRINT,
//        DISP=(NEW,CATLG,DELETE),
//        DCB=(RECFM=LSEQ,LRECL=256)
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
  DELETE                                -
     INCLUDE(TEST.FB.FILE*.ABC)         -
 CANCEL
/*
 //
```

RESTORE工作

此工作將從TESTDUMP備份位置還原與INCLUDE參數匹配的文件。如果目錄檔案與CATDUMP匯出中的檔案相同，且已指定REPLACE選項，則會取代已編目的檔案。

```
//M2DFREST JOB 'M2DFREST',CLASS=A,MSGCLASS=X
//STEP001  EXEC PGM=M2DFUTIL
////SYSPRINT DD DSN=TESTREST.SYSPRINT,
//        DISP=(NEW,CATLG,DELETE),
//        DCB=(RECFM=LSEQ,LRECL=256)
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
RESTORE SOURCE(TESTDUMP)               -
     INCLUDE(TEST.FB.FILE*.ABC)        -
 IGNORE
 REPLACE
```

```
/*
//
```

M2 RUNCMD 批次工具

您可以使用批次公用程式 M2RUNCMD，直接從執行 Micro Focus 命令、指令碼和系統呼叫，JCL而不是從終端機或命令提示字元執行這些命令、指令碼和系統呼叫。命令的輸出會記錄到批次工作的多工緩衝處理記錄中。

主題

- 支援平台
- 在 Amazon 上RUNCMD為 AWS 大型主機現代化運行時配置 M2EC2（包括 2.0） AppStream
- 樣品 JCLs

支援平台

您可以在以下平台RUNCMD上使用 M2：

- 微焦點運行時（在 Amazon 上EC2）
- 微焦點企業開發商（ED）和微焦點企業服務器（ES）產品的所有變體。

在 Amazon 上RUNCMD為 AWS 大型主機現代化運行時配置 M2EC2（包括 2.0） AppStream

如果您遷移的應用程式在 Amazon 上的 AWS 大型主機現代化執行階段上執行EC2，請按如下方式設定 M2 RUNCMD。

- 變更 Micro Focus JES 程式路徑以包含批次公用程式的二進位位置。如果您必須指定多個路徑，請在 Linux 上使用冒號 (:) 分隔路徑，在 Windows 上使用分號 (;) 來分隔路徑。
  - Linux：`/opt/aws/m2/microfocus/utilities/64bit`
  - 視窗 (32 位元): `C:\AWS\M2\MicroFocus\Utilities\32bit`
  - 視窗 (64 位元): `C:\AWS\M2\MicroFocus\Utilities\64bit`

樣品 JCLs

若要測試安裝，您可以使用下列其中一個範例JCLs。

RUNSCRL1.jcl

此範例JCL會建立指令碼並執行它。第一個步驟會建立名為/tmp/TEST_SCRIPT.sh並包含SYSUT1串流內資料內容的指令碼。第二個步驟會設定執行權限，並執行在第一個步驟中建立的指令碼。您也可以選擇只執行第二個步驟來執行現有的 Micro Focus 和系統指令。

```
//RUNSCRL1 JOB 'RUN SCRIPT',CLASS=A,MSGCLASS=X,TIME=1440
//*
//*
//*--------------------------------------------------------------------*
//*  CREATE SCRIPT (LINUX)
//*--------------------------------------------------------------------*
//*
//STEP0010 EXEC PGM=IEBGENER
//*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//*
//SYSUT1   DD *
#!/bin/bash

set -x

## ECHO PATH ENVIRONMNET VARIABLE
echo $PATH

## CLOSE/DISABLE VSAM FILE
casfile -r$ES_SERVER -oc  -ed -dACCTFIL

## OPEN/ENABLE VSAM FILE
casfile -r$ES_SERVER -ooi -ee -dACCTFIL

exit $?
/*
//SYSUT2   DD DSN=&&TEMP,
//            DISP=(NEW,CATLG,DELETE),
//            DCB=(RECFM=LSEQ,LRECL=300,DSORG=PS,BLKSIZE=0)
//*MFE: %PCDSN='/tmp/TEST_SCRIPT.sh'
//*
//*--------------------------------------------------------------------*
//*   RUN SCRIPT (LINUX)                                               *
//*--------------------------------------------------------------------*
//*
```

```
//STEP0020 EXEC PGM=RUNCMD
//*
//SYSOUT  DD  SYSOUT=*
//*
//SYSIN   DD *
*RUN SCRIPT
 sh /tmp/TEST_SCRIPT.sh
/*
//
```

SYSOUT

執行的命令或指令碼的輸出會寫入SYSOUT記錄中。對於每個執行的命令，它會顯示命令，輸出和返回代碼。

```
************ CMD Start ************

CMD_STR: sh /tmp/TEST_SCRIPT.sh

CMD_OUT:


+ echo /opt/microfocus/EnterpriseServer/bin:/sbin:/bin:/usr/sbin:/usr/bin
/opt/microfocus/EnterpriseServer/bin:/sbin:/bin:/usr/sbin:/usr/bin
+ casfile -rMYDEV -oc -ed -dACCTFIL

-Return Code:   0

Highest return code:    0

+ casfile -rMYDEV -ooi -ee -dACCTFIL

-Return Code:   8

Highest return code:    8

+ exit 8


CMD_RC=8


************  CMD End  ************
```

RUNCMDL1.jcl

本示例JCL用RUNCMD於運行多個命令。

```
//RUNCMDL1 JOB 'RUN CMD',CLASS=A,MSGCLASS=X,TIME=1440
//*
//*
//*--------------------------------------------------------------------*
//*   RUN SYSTEM COMMANDS                                              *
//*--------------------------------------------------------------------*
//*
//STEP0001 EXEC PGM=RUNCMD
//*
//SYSOUT  DD  SYSOUT=*
//*
//SYSIN   DD *
*LIST DIRECTORY
 ls
*ECHO PATH ENVIRONMNET VARIABLE
 echo $PATH
/*
//
```

AWS 精確運用大型主機現代化資料複製

AWS 大型主機現代化提供各種 Amazon 機器映像 ()。AMIs這有AMIs助於快速佈建 Amazon EC2 執行個體，為從大型主機系統複寫到 AWS 使用的資料建立量身打造的環境。本指南提供存取和使用這些步驟所需的步驟AMIs。

必要條件

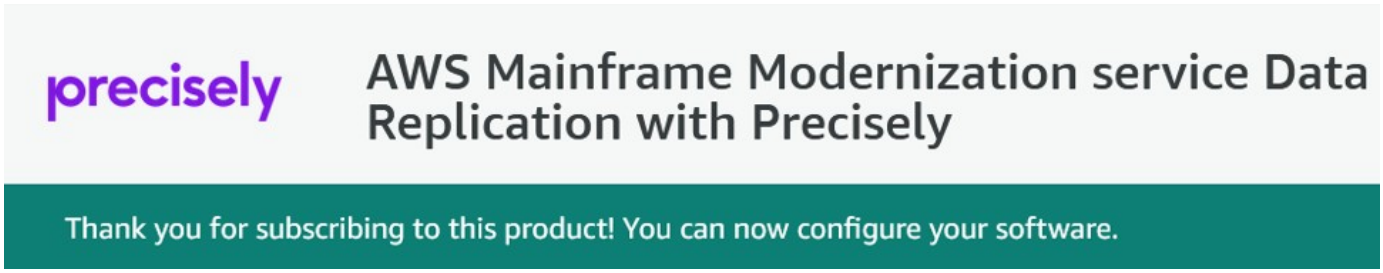
- 確保您擁有可以建立 Amazon EC2 執行個體的 AWS 帳戶的管理員存取權。
- 確認您計劃建立 Amazon 執行個體的區域中是否提供 AWS 大型主機現代化服務。EC2請參閱按地區提供的AWS服務清單。
- 識別將在其中建立 Amazon EC2 執行個體的 Amazon Virtual Private Cloud (AmazonVPC)。
- 在 Amazon 中建立 Amazon EC2 執行個體時VPC，請確保相關聯的路由表具有網際網路閘道或NAT閘道。

> **Note**
>
> 成功複製資料需要AWSEC2執行個體具有 AWS Marketplace 的通訊存取權。如果 Mar AWS ketplace 發生連線問題，複寫程序將會失敗。

訂閱 Amazon 機器映像

當您訂閱 AWS Marketplace 產品時，您可以從產品的執行個體啟動執行個體AMI。

1. 登入 AWS Management Console 並在 https://console.aws.amazon.com/Marketplace 開啟 AWS Marketplace 主控台。
2. 選擇 Manage subscriptions (管理訂閱)。
3. 將以下鏈接複製並粘貼到瀏覽器地址欄中 https://aws.amazon.com/marketplace/：
4. 選擇 Continue to Subscribe (繼續以訂閱)。
5. 如果條款和條件可接受，請選擇「接受條款」。訂閱可能需要幾分鐘的時間來處理。
6. 等待感謝信息出現，如下所示。此訊息確認您已成功訂閱產品。



7. 在左側導覽窗格中，選擇 [管理訂閱]。此檢視會顯示您已訂閱的所有訂閱。

使用 AWS 精確地啟動大型主機現代化資料複製

1. 在 https://console.aws.amazon.com/市集開啟 AWS Marketplace 主控台。
2. 在左側導覽窗格中，選擇 [管理訂閱]。
3. 尋找您AMI要啟動的項目，然後選擇 [啟動新執行個體]。
4. 在「區域」下，選取允許列出的區域。
5. 選擇 [繼續] 以透過啟動EC2。此操作將帶您到 Amazon EC2 控制台。
6. 輸入伺服器的名稱。
7. 選取符合您專案效能和成本需求的執行個體類型。例證大小的建議起點為`c5.2xLarge`。
8. 選擇現有的 key pair，或建立並儲存新的金鑰配對。如需金鑰配對的相關資訊，請參閱 Amazon EC2 使用者指南中的 Amazon EC2 金鑰配對和 Linux 執行個體。
9. 編輯網路設定，並選擇允許列出VPC和適當的子網路。
10. 選擇現有的安全性群組或建立新群組。除了允許SSH存取 (預設為連接埠 22) 之外，對於使用精確伺服器EC2執行個體進行資料複製，通常允許TCP流量傳輸至其預設連接埠 2626。
11. 設定 Amazon EC2 執行個體的儲存空間。
12. 檢閱摘要，然後選擇啟動執行個體。若要成功啟動，執行個體類型必須是有效的。如果啟動失敗，請選擇 [編輯執行個體組態] 並選擇不同的執行個體類型。
13. 看到成功訊息後，請選擇 [Connect 至執行個體]。
14. 在打開 Amazon EC2 控制台https://console.aws.amazon.com/ec2/。
15. 在左側導覽窗格的 [執行個體] 功能表下，選擇 [執行個體]。
16. 在主窗格中，檢查執行個體的狀態。

建立IAM策略

若要成功操作透過 AWS 我們的 AWS Marketplace 清單部署的大型主機現代化EC2執行個體，您必須設定IAM角色和政策。此特IAM定量身打造的設定並非選擇性設定；它會授權您的 Amazon EC2 執行個體與服務互動。 AWS Marketplace IAM角色和原則可 AWS 讓大型主機現代化精確記錄使用情況資料，這對於精確計費至關重要。無法實作此組態可能會導致資料複寫嘗試失敗和作業中斷。

1. 在開啟IAM主控台https://console.aws.amazon.com/iam/。
2. 在左側的導覽窗格中，選擇 Policies (政策)。
3. 如果這是您第一次選擇原則，[歡迎使用受管理的策略] 頁面隨即出現。選擇開始使用。
4. 在頁面頂端，選擇 Create policy (建立政策)。
5. 在 [原則編輯器] 區段中，選擇JSON選項。
6. 輸入下列JSON策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": ["aws-marketplace:MeterUsage"],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

建立 IAM 角色

1. 在開啟IAM主控台https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。
3. 在 Trusted entity type (信任的實體類型) 區段中，選擇 AWS service (服務)。
4. 在「使用案例」區段的「服務」或「使用案例」下，選擇 Amazon EC2。
5. 選擇 Next (下一步)。
6. 在策略清單中，從「依類型篩選」下拉式清單中選取「客戶管理」，然後輸入您建立的策略名稱。選取策略名稱旁邊的核取方塊。
7. 選擇 Next (下一步)。

8. 輸入角色的名稱，並選擇性地輸入說明。
9. 檢閱信任原則和權限，然後選擇 [建立角色]。

將IAM角色附加到 Amazon 執EC2行個體

1. 在打開 Amazon EC2 控制台https://console.aws.amazon.com/ec2/。
2. 在導覽窗格中，選擇 Instances (執行個體)。
3. 選擇您的 Amazon EC2 實例。
4. 從 [動作] 功能表中選擇 [安全性]，然後選擇 [修改IAM角色]。
5. 選取要附加至執行個體的角色，然後選擇 [更新IAM角色]。

AWS Mainframe Modernization 代碼轉換 mLogica

AWS Mainframe Modernization 程式碼轉換 mLogica (程式碼轉換) 是自動將 z/OS 大型主機彙編程式程式碼轉換為的 AWS Mainframe Modernization 功能。COBOL您可以使用程式碼轉換，使用 AWS CodeBuild 服務來提取彙編程式影像，以便與. AWS 帳戶

主題

- 彙編程序轉換是什麼？mLogica
- 瞭解彙編程式轉換的代碼轉換帳單
- 程式碼轉換概念
- 瞭解程式碼轉換的元件和程序
- 教學課程：將程式碼從彙編程式轉換為 in COBOL AWS Mainframe Modernization

彙編程序轉換是什麼？mLogica

AWS Mainframe Modernization 使用 mLogica (程式碼轉換) 的程式碼轉換會自動將 z/OS 大型主機彙編程式碼轉換為. COBOL 該服務在您 AWS 帳戶 的內部運行，並且不會在 AWS 帳戶. COBOL 代碼轉換允許您的授權帳戶使用該 AWS CodeBuild 服務提取彙編程序映像以進行預期的代碼轉換。

AWS Mainframe Modernization 讓您能夠為已移轉的應用程式設定組建和持續整合/持續交付 (CI/CD) 管線。這些組建和管道使用 AWS CodeBuild Amazon S3 來提供此功能。 AWS CodeBuild 是完全受控的建置服務，可編譯原始程式碼、執行單元測試，以及產生準備好部署的成品。Amazon S3 是一種物件儲存服務，提供領先業界的可擴展性、資料可用性、安全性和效能。

主題

- 代碼轉換編譯器
- 程式碼轉換架構
- 自動化方法
- 安全
- 其他資源

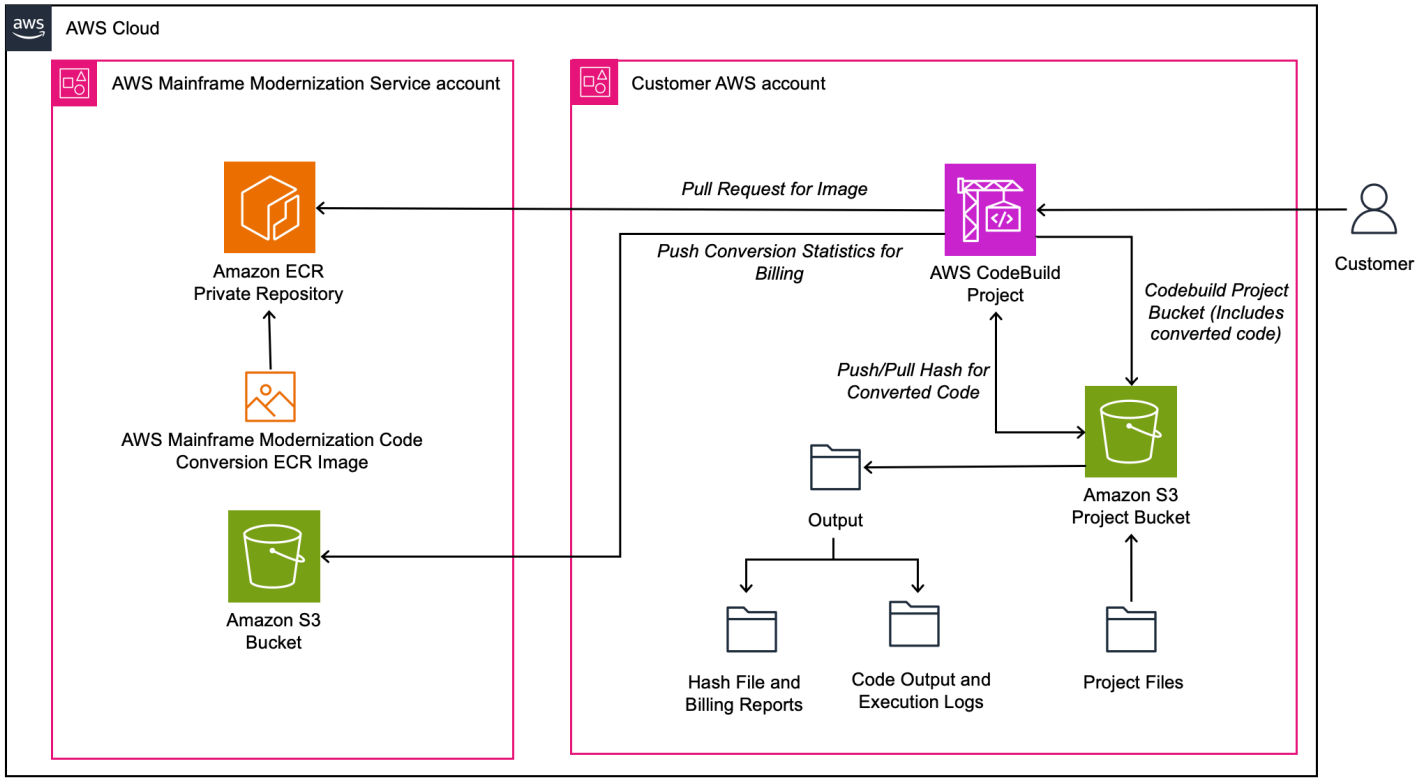
代碼轉換編譯器

程式碼轉換可以設定為發出COBOL適合在具有不同編譯器的多個目標環境中進行編譯和執行。其中一些包括：

- M2 利用微焦點和其他微焦點企業伺服器環境重新平台
- M2 與NTTDATA企業COBOL重新平台 () UniKix
- mLogica LIBER*COBOL
- 使用企業級的 z/OS 大型主機 IBM COBOL
- 非常 COBOL

程式碼轉換架構

以下是程式碼轉換程序的架構圖：



自動化方法

若要搭配使用程式碼轉換 CodeBuild，需要將彙編程式碼上傳到 Amazon S3 儲存貯體，以便稍後設定轉換參數，並叫用 CodeBuild 專案以執行轉換程序中的每個步驟。目標COBOL代碼會自動存放在 Amazon S3 儲存貯體中的指定路徑中。

安全

AWS Mainframe Modernization 代碼轉換可實現轉換，同時將所有源代碼和目標代碼保留在 AWS 帳戶. 來源彙編程式程式碼、目標COBOL程式碼和組態檔案都存放在 Amazon S3 儲存貯體中。自動轉換工具會在您的 CodeBuild 環境中以容器的形式執行 AWS 帳戶。該代碼始終保留在您的帳戶中。

若要啟用轉換工具存取 Amazon S3 儲存貯體，請將存取儲存貯體的許可授予 AWS 服務 角色。設定時 CodeBuild，您將設定此服務角色， CodeBuild 以便存取容器映像並存取 Amazon S3 儲存貯體。

其他資源

除了the section called “教學課程：將程式碼從彙編程式轉換為 COBOL”，以下是一些額外的資源，您可以在其中瞭解如何建立 AWS CloudFormation 範本，以及有關將彙編程式轉換為的其他資訊。COBOL

- 工作坊連結，可將自動程式碼從彙編程式轉換為COBOL：https://catalog.workshops.aws/awsm2ccm-assembler-cobol/en-US。
- 博客文章：https://aws.amazon.com/blogs/migration-and-modernization/unlocking-new-potential-transform-your-assembler-programs-to-cobol-with-aws-mainframe-modernization/.

瞭解彙編程式轉換的代碼轉換帳單

在進行實際轉換之前，您將參考此頁面以了解代碼轉換計費範圍和過程。計費計算部分提到了從彙編程序轉換COBOL為按每行代碼收取的過程。

代碼轉換計費和範圍

只有在完成轉換步驟後，組裝程式代碼轉換 AWS 帳戶 才會產生費用 (帳單報表)。費用是根據轉換的代碼行數而定。如果您執行多個轉換步驟，例如在新增新的 Assembler 程式碼、變更轉換組態或套用新版本的貨櫃之後，只會使用變更的明細行和/或新增的明細行來計算費用。我們不會向您收取兩次同一行代碼在同一程序中轉換費用。

> **Note**
>
> 具有更改代碼行的模塊以及新的或重命名的程序中的所有代碼行將被收取費用。

為了避免多重費用，程式碼轉換會為中的專案值區中的每個彙編程式或 Macro 模組儲存一個編碼的二進位檔案。`<Project_bucket>/awsm2ccm-do-not-delete/<AWS_account_number>/Hash`這些編碼的文件不包含任何客戶代碼。

> **Important**
>
> 請勿手動編輯或刪除這些檔案。變更可能會導致轉換相同元件的多個帳單。

程 AWS Mainframe Modernization 式碼轉換分析報告 (以下稱「分析報告」) 為客戶提供預期轉換範圍、結果和帳單的詳細資訊，以確保對實際轉換的準確預期。轉換可能會導致某些代碼行無法轉換，某些代碼行部分被轉換，以及一些代碼行完全轉換。「分析報告」會顯示每個品類的程式碼行數。您必須先執行並閱讀「分析報告」，才能處理程式、巨集和撰寫本的任何轉換。一旦客戶複查了「分析報告」並同意報告的範圍，預期結果和預期的帳單，客戶就可以繼續執行轉換。

> **Note**
>
> 透過執行AWS大型主機現代化程式碼轉換**`Convert`**命令，即表示您確認已執行並閱讀分析報告，並同意預期的結果和可計費的程式碼行數。

轉換範圍

AWS Mainframe Modernization 程式碼轉換會處理設定 S3 來源位置中 scrlib 和 macrolib 目錄中所有彙編程式、巨集和文字本元件的所有程式碼行。彙編程序，以及彙編程序中引用的任何宏和字帖都在範圍內。未被彙編程序引用的宏和抄本組件被認為是超出範圍，而不是轉換。在處理過程中，轉換器會執行進階演算法，以整體考量範圍內的每個元件。這些元件的所有程式碼行都會參與處理，無論這些元件是否已完全轉換、部分轉換或未轉換。 AWS Mainframe Modernization 代碼轉換會忽略空行，並且不會將它們計為代碼行。包含任何其他文字的註解行和行 (例如，內嵌於中的彙編程式JCL陳述式JCL) 會計為計費程式碼行。

計費計算

AWS Mainframe Modernization 整個範圍內元件的程式碼轉換費用。這表示它會針對每個範圍內元件中的每一行程式碼收費，包括無法轉換、部分轉換並完全轉換的行。 AWS Mainframe Modernization 程式碼轉換會將提供給處理之元件的所有程式碼行加總 (包括彙編程式、參考文獻和參考的巨集)，並使用程式碼的總行數來計費。

> **Note**
>
> 彙編程序未引用的字稿和宏不被視為範圍內。

例如，假設一個程式有 1,000 行程式碼：

- 完全轉換 700 條線
- 200 行被部分轉換
- 100 行未轉換

1,000 行代碼將被處理，並將被計費。

改善轉換

如果您身為客戶尋求更高的程式碼行轉換率，或有其他特定需求，您可以與 AWS 代表聯絡，以取得其他參與選項，例如校正工作或專業服務協助。

程式碼轉換概念

要了解代碼轉換是如何發生的，請了解一些關鍵概念，例如宏處理，代碼頁，這一點 CodeBuild 很重要。

主題

- 巨集處理
- 字碼頁 (EBCDICvsASCII)
- CodeBuild

巨集處理

大型主機彙編程式碼經常使用巨集來封裝功能以供重複使用。宏的行為通常是在應用程序運行時根據從彙編程序傳遞的參數確定的。代碼轉換提供了幾種機制，用於在轉換為之前擴展彙編程序宏。COBOL

字碼頁 (EBCDICvsASCII)

大型主機彙編程序通常包含字符文字表示為對應於字符的十六進制值。EBCDIC程式碼轉換提供可設定的功能，以便在發出COBOL環境ASCII時自動管理中的字元常值。ASCII

CodeBuild

代碼轉換可通過該 AWS CodeBuild 服務進行。 AWS CodeBuild 是最初設計為 CI/CD 管道的一部分的構建自動化工具。在中 AWS Mainframe Modernization，用 AWS CodeBuild 於自動化轉MCCAC換工具和其他工具，例如 Micro Focus COBOL 編譯器。

瞭解程式碼轉換的元件和程序

AWS Mainframe Modernization 程式碼轉換程序包括各種元件，例如 AWS Mainframe Modernization 容器、S3 專案儲存貯體和日誌檔位置。

主題

- AWS Mainframe Modernization 容器
- S3 專案儲存貯體
- 記錄檔位置
- 程序概觀

AWS Mainframe Modernization 容器

AWS Mainframe Modernization 程式碼轉換容器會在 AWS CodeBuild 專案中執行，並提供指令來設定專案目錄和組態檔、評估彙編程式碼、展開彙編程式巨集，以及將彙編程式碼轉換為。COBOL

您將可以訪問以下AWSECR存儲庫：`381492161314.dkr.ecr.us-east-1.amazonaws.com/aws-mlogica-codebuild-prod`。

要使用圖像，您可以按照以下兩個選項之一進行操作：

- 透過使用影像時，請使用最新的標籤 AWS CodeBuild。使用圖像時，您將使用以下路徑：381492161314.dkr.ecr.us-east-1.amazonaws.com/aws-mlogica-codebuild-prod。這意味著 AWS CodeBuild 將拿起最後一個推送的映像到存儲庫中的哪個。
- 列出版本並從中選擇。要執行此操作，請使用以下命令 via CLI 列出存儲庫中的不同版本：

```
aws ecr describe-images \
  --registry-id 381492161314 \
  --repository-name aws-mlogica-codebuild-prod \
  --query 'imageDetails[*].{ImagePushedAt: imagePushedAt, ImageTags: imageTags}' \
  --output json | jq '[.[] | {ImageURI: (.ImageTags[] |
 "381492161314.dkr.ecr.us-east-1.amazonaws.com/aws-mlogica-codebuild-prod:" + .),
 ImagePushedAt: .ImagePushedAt}] | sort_by(.ImagePushedAt) | reverse'
```

這會列出每個影像上有關聯標記的所有影像，以及將特定影像釋放至儲存庫的時間。基於上面的代碼，您將獲得圖像列表，其中圖像上的標籤表示代碼轉換實用程序的版本。您可以根據自己的需求選擇適當的圖像。

S3 專案儲存貯體

輸入和輸出代碼，使用擴展的 Macros 更新的代碼以及 AWS Mainframe Modernization 代碼轉換生成的報告都存儲在您在 AWS Account Management. 您可以透過授與 AWS 服務角色的權限來提供 AWS Mainframe Modernization 程式碼轉換以存取值區的權限。

記錄檔位置

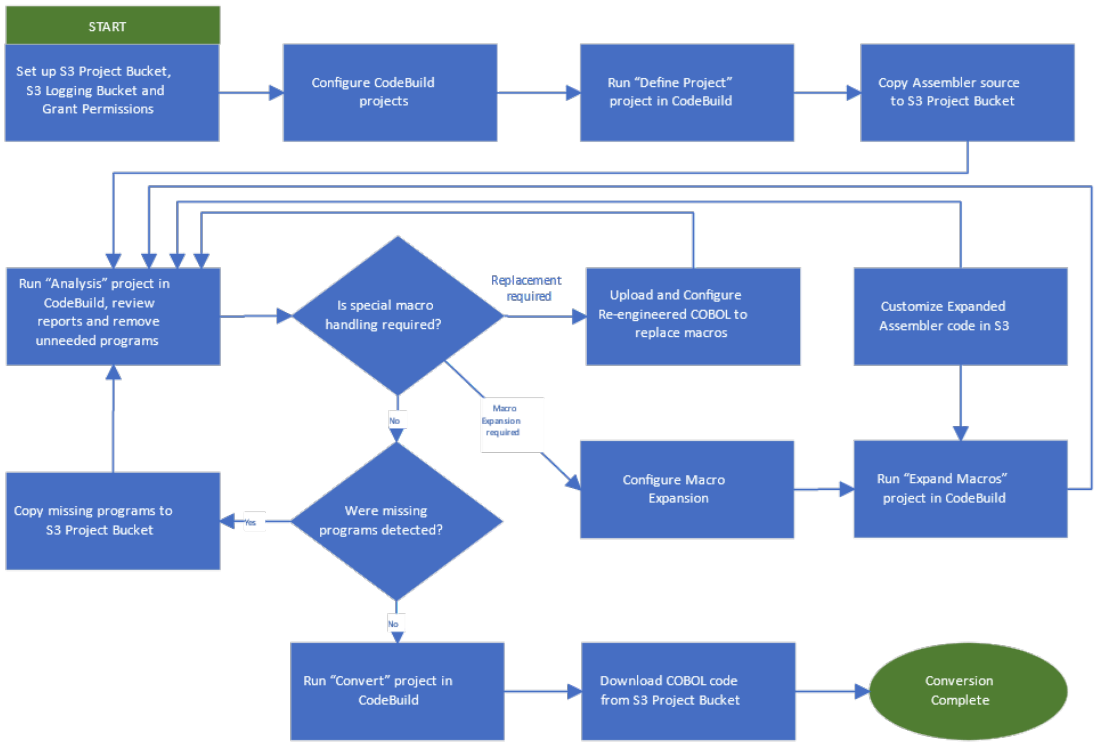
在每個 CodeBuild 專案執行期間，記錄檔會寫入兩個位置：

- 具有每個 CodeBuild 步驟之高階結果的記錄檔會寫入至中設定之 Logging 值區中的記錄檔 CodeBuild。這些文件顯示為 gzip 歸檔，其中包含 CodeBuild 框架生成的 GUID-類型文件名（例如，0c03e183-ab40-4fe0-ba77- bc1d87e73b14.gz）。每個歸檔包含 CodeBuild 專案執行所產生的記錄檔。如果 CodeBuild專案執行失敗，此記錄檔將包含重要的疑難排解資訊。
- 在元件層級具有詳細執行結果的記錄檔會以檔案名稱模式 <Project_Bucket_name>_.log (例如project- bucket_202406131200.log) 寫入主要 Project 值區路徑中的記錄檔。這些記錄檔提供：
  - 說明輸入和輸出位置的組態摘要。
  - 使用目標檔案名稱處理的每個彙編程式或巨集元件的記錄。
  - 使用檔案位置產生的報告清單。

- 對於轉換執行，則提供了運行時期抄本的列表。

程序概觀

下圖說明了將彙編程序轉換為的過程：COBOL



教學課程：將程式碼從彙編程式轉換為 in COBOL AWS Mainframe Modernization

您可以使用本文件做為 step-by-step 指南，以瞭解如何將大型主機現代化彙編程式碼轉換為. COBOL 除此之外，您還可以參考從彙編程序到COBOL研討會的自動代碼轉換，以了解有關轉換過程的更多信息。

主題

- 必要條件
- 步驟 1：共用組建資產 AWS 帳戶
- 步驟 2：創建 Amazon S3 存儲桶
- 步驟 3：建立IAM策略

- 步驟 4：建立IAM角色
- 步驟 5：將IAM策略附加到IAM角色
- 步驟 6：建立 CodeBuild 專案
  - 步驟 6.1：創建定義項目
  - 步驟 6.2：建立程式碼分析專案
  - 步驟 6.3：創建代碼轉換項目
- 第 7 步：定義項目並上傳源代碼
- 步驟 8：執行分析並瞭解報告
- 步驟 9：執行程式碼轉換
- 步驟 10：驗證代碼轉換
- 步驟 11：下載轉換後的代碼
- 清除資源

必要條件

請閱讀本瞭解彙編程式轉換的代碼轉換帳單章節，瞭解 Assembler 程式碼轉換如何產生您的費用 (帳單報告) AWS Account Management，以及帳單運作方式。

步驟 1：共用組建資產 AWS 帳戶

在此步驟中，請務必與您的共用組建資產 AWS 帳戶，尤其是在使用資產的地區。

1. 在開啟 AWS Mainframe Modernization 主控台https://console.aws.amazon.com/m2/。
2. 在左側導覽中，選擇 [工具]。
3. 在AWS大型主機現代化程式碼轉換中mLogica，選擇 [與我的. AWS 帳戶

> **Important**
>
> 您需要在您打算進行構建的每個 AWS 區域中執行此步驟一次。

步驟 2：創建 Amazon S3 存儲桶

在此步驟中，您會建立 Amazon S3 儲存貯體。第一個存儲桶是用於保存源代碼 AWS CodeBuild 的項目存儲桶，然後推送輸出存儲桶以保存 AWS CodeBuild 輸出（轉換後的代碼）。如需詳細資訊，請參閱 Amazon S3 使用者指南中的建立、設定和使用 Amazon S3 儲存貯體。

1. 若要建立專案儲存貯體，請登入 Amazon S3 主控台，然後選擇「建立儲存貯體」。
2. 在 [一般] 設定中，提供值區的名稱，並指定您 AWS 區域 要建立值區的位置。一個例子名稱是*codebuild-regionId-accountId-bucket*，其中：
   - `regionId`是桶 AWS 區域 的。
   - `accountId`是您的 AWS 帳戶 身份證。

   > **Note**
   >
   > 如果您在不同 AWS 區域 於美國東部 (維吉尼亞北部) 建立值區，請指定`LocationConstraint`參數。如需詳細資訊，請參閱 Amazon 簡單儲存服務API參考中的建立儲存貯體。
3. 保留所有其他設定，然後選擇 [建立值區]。

無論您為這些值區選擇什麼名稱，請務必在本教學課程中使用它們。

步驟 3：建立IAM策略

在此步驟中，您會建立IAM原則。提供的IAM政策授予 AWS CodeBuild 與 Amazon S3、Amazon 彈性容器登錄、 CodeBuild 產生的 Amazon CloudWatch 日誌以及用於程式碼轉換的 Amazon Elastic Compute Cloud 資源互動的特定許可。此政策不是為客戶量身定制的。該政策授予互動的 AWS Mainframe Modernization 權限，並獲取代碼轉換統計信息，以適當地向客戶收取費用。

若要瞭解如何建立IAM策略，請參閱IAM使用指南中的建立IAM策略。

若要建立策略

1. 登入IAM主控台，然後在左側導覽窗格中選擇 [原則]。
2. 選擇 建立政策。
3. 將以下JSON策略複製並粘貼到策略編輯器中。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:GetBucketLocation",
                "s3:ListBucket",
                "s3:PutObjectAcl",
                "s3:GetBucketAcl"
            ],
            "Resource": [
                "arn:aws:s3:::codebuild-regionId-accountId-bucket",
                "arn:aws:s3:::codebuild-regionId-accountId-bucket/*",
                "arn:aws:s3:::aws-m2-repo-*" ],
                    "Effect": "Allow"
          },
          {
              "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer",
                "logs:*",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DeleteNetworkInterface",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeVpcs",
                "ec2:CreateNetworkInterfacePermission"
            ],
            "Resource": "*",
            "Effect": "Allow"
         }
    ]
}
```

4. 您可以選擇性地將標籤新增至策略。標籤是索引鍵值配對，可協助您組織、追蹤或控制原則的存取權。

5. 選擇 Next:Review (下一步：檢閱)。
6. 提供原則的名稱，例如*CodeBuildAWSM2CCMPolicy*。
7. 您可以選擇性地輸入策略的描述，然後檢閱策略摘要以確保原則正確無誤。
8. 選擇 建立政策。

步驟 4：建立IAM角色

在此步驟中，您會建立一個新IAM角色， CodeBuild 以便在您先前建立的IAM原則與此新IAM角色建立關聯之後，為您與 AWS 資源互動。

如需有關建立服務角色的資訊，請參閱IAM使用指南中的建立角色以將權限委派給 AWS 服務。

1. 登入IAM主控台，然後在左側導覽窗格中選擇 [角色]。
2. 選擇建立角色。
3. 在 [信任的實體類型] 下，選擇 [AWS服務
4. 在「其他AWS服務的使用案例」下 CodeBuild，選擇，然後CodeBuild再次選擇。
5. 選擇 Next (下一步)。
6. 在 Add permissions (新增許可) 頁面上，選擇 Next (下一步)。稍後您可以將原則指派給角色。
7. 在「角色詳細資料」下，提供角色的名稱，例如，`IAMRoleTaskExecutionRoleForCodeBuild`。
8. 在 [選取信任的實體] 底下，確認原則文件如下所示：

```
{
          "Version": "2012-10-17",
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": "codebuild.amazonaws.com"
              },
              "Action": "sts:AssumeRole"
            }
          ]
}
```

9. 選擇建立角色。

步驟 5：將IAM策略附加到IAM角色

在此步驟中，您會將先前建立的IAM原則附加至IAMRoleTaskExecutionRoleForCodeBuildIAM角色。

1. 登入IAM主控台，然後在左側導覽窗格中選擇 [角色]。
2. 在「角色」中，選擇您先前建立的角色，例如，IAMRoleTaskExecutionRoleForCodeBuild。
3. 在 [權限] 原則中，選擇 [新增權限]，然後選擇 [附加原則
4. 在 [其他權限原則] 中，選擇您先前建立的原則，例如*CodeBuildAWSM2CCMPolicy*。
5. 選擇連接政策。

步驟 6：建立 CodeBuild 專案

在此步驟中，您會根據上述buildspec.yml檔案建立三個不同的 CodeBuild 專案。

步驟 6.1：創建定義項目

若要建立定義專案

1. 登入 CodeBuild 主控台，然後選擇 [建立組建專案]。
2. 在 [專案組態] 區段中，提供專案的名稱，例如1-awsm2ccm-define-project。
3. 在 [來源] 區段中，對於 [來源提供者]，保留預設選取項目。
4. 在「環境」區段中，選擇「自訂影像」。
5. 在「環境類型」欄位中，選擇「Linux」。
6. 在 [影像登錄] 下，選擇 [其他登錄]
7. 在 [外部登錄 URL] 欄位中，遵循下列the section called “AWS Mainframe Modernization 容器”章節。
8. 在 [服務角色] 底下，選擇 [現有服務角色]，然後在 [角色 ARN] 欄位中選擇您先前建立的服務角色(例如IAMRoleTaskExecutionRoleForCodeBuild)。
9. 展開 [其他組態] 區段，執行下列動作：
   a. VPC：根據您的設置進行配置（如果需要）。
   b. 逾時：設定為 60 分鐘。
   c. 佇列逾時：設定為 480 分鐘。

d. 加密：選擇適當的加密設定 (預設值沒問題)。

e. 在「環境變數」區段中，逐一新增下列項目：

- 名稱:PROJECT_ BUCKET. 值：**codebuild-regionId-accountId- bucket**。類型：明文
- 名稱:PROJECT_ DIR. 值：**prj_codebuild_01**。類型：明文
- 名稱:AWSM2CCM_ ACTION. 值：**define_project**。類型：明文
- 名稱：AWSM2CCM_ LOGGING _ BUCKET. 值：**s3:// codebuild-regionId-accountId-bucket**。類型：明文

10. 在 [Buildspec] 區段中，選擇 [插入建置命令]，然後選擇 [切換至編輯器]。
11. 用以下內容替換當前值：

```
version: 0.2
phases:
    build:
        commands:
            - . /app/awsm2ccm_prod/bin/setup_env.sh
            - run_awsm2ccm.sh $PROJECT_DIR
artifacts:
    files:
        - '**/*'
    discard-paths: no
    base-directory: $PROJECT_DIR
```

其中，PROJECT_ DIR 是可用的環境變量 CodeBuild。如需詳細資訊，請參閱建置環境中的環境變數。

12. 在「成品」區段中，執行下列動作：

- 在「類型」下，選擇 Amazon S3，然後選擇您的輸出儲存貯體，例如codebuild-regionId-accountId-bucket。
- 若為「路徑」，請將此欄位保留空白。
- 對於「名稱」，輸入**prj_codebuild_01**。
- 對於 Artifact 封裝，選取無。
- 針對覆寫人工因素名稱，取消勾選此選項。
- 對於加密，請將其保留為默認設置。

13. 針對「記錄檔」區段，執行下列動作：

- CloudWatch 記錄檔：已停用
- S3 日誌：已啟用

- 鏟斗:**codebuild-regionId-account-bucket**
- 記錄檔路徑：**CODEBUILD-LOGS**

14. 選擇 Create build project (建立建置專案)。

步驟 6.2：建立程式碼分析專案

若要建立程式碼分析專案

1. 登入 CodeBuild 主控台，然後選擇 [建立組建專案]。
2. 在 [專案組態] 區段中，提供專案的名稱，例如`2-awsm2ccm-analysis`。
3. 在 [來源] 區段中，針對 [來源提供者] 選擇 Amazon S3，然後選擇您先前建立的輸入儲存貯體 (例如`codebuild-regionId-accountId-bucket`)。
4. 在 S3 物件金鑰或 S3 資料夾欄位中，輸入**prj_codebuild_01**。
5. 在「環境」區段中，選擇「自訂影像」。
6. 在「環境類型」欄位中，選擇「Linux」。
7. 在 [影像登錄] 下，選擇 [其他登錄]
8. 在 [外部登錄 URL] 欄位中，遵循下列the section called “AWS Mainframe Modernization 容器”章節。
9. 在 [服務角色] 底下，選擇 [現有服務角色]，然後在 [角色 ARN] 欄位中選擇您先前建立的服務角色 (例如`IAMRoleTaskExecutionRoleForCodeBuild`)。
10. 展開 [其他組態] 區段，執行下列動作：

    a. VPC：根據您的設置進行配置（如果需要）。

    b. 逾時：設定為 60 分鐘。

    c. 佇列逾時：設定為 480 分鐘。

    d. 加密：選擇適當的加密設定 (預設值沒問題)。

    e. 在「環境變數」區段中，逐一新增下列項目：

    - 名稱:PROJECT_ BUCKET. 值：**codebuild-regionId-accountId-bucket**。類型：明文

- 名稱:PROJECT_ DIR. 值：**prj_codebuild_01**。類型：明文
- 名稱:AWSM2CCM_ ACTION. 值：**analysis**。類型：明文
- 名稱：AWSM2CCM_ LOGGING _ BUCKET. 值：**s3:// codebuild-regionId-accountId-bucket**。類型：明文

11. 在 [Buildspec] 區段中，選擇 [插入建置命令]，然後選擇 [切換至編輯器]。
12. 用以下內容替換當前值：

```
version: 0.2
phases:
    build:
        commands:
            - ln -s $CODEBUILD_SRC_DIR $PROJECT_DIR
            - . /app/awsm2ccm_prod/bin/setup_env.sh
            - run_awsm2ccm.sh $PROJECT_DIR
artifacts:
    files:
        - '*.log'
        - '_Converted/*/*'
        - '_Reports/*'
    secondary-artifacts:
        reports:
            files:
                - '_Reports/AWSM2CCM*'
    discard-paths: no
    base-directory: $PROJECT_DIR
```

其中，PROJECT_ DIR 是可用的環境變量 CodeBuild。如需詳細資訊，請參閱建置環境中的環境變數。

13. 在「成品」區段中，執行下列動作：

- 在 [類型] 下，選擇 Amazon S3，然後選擇您的輸出儲存貯體 (例如codebuild-regionId-accountId-bucket)。
- 輸入做為「路徑」ARTIFACTS。
- 對於「名稱」，輸入**prj_codebuild_01**。
- 對於 Artifact 封裝，選取無。
- 針對覆寫人工因素名稱，取消勾選此選項。
- 對於加密，請將其保留為默認設置。

14. 針對「記錄檔」區段，執行下列動作：

    - CloudWatch 記錄檔：已停用
    - S3 日誌：已啟用

    - 鏟斗:**codebuild-regionId-account-bucket**
    - 記錄檔路徑：**CODEBUILD-LOGS**

15. 選擇 Create build project (建立建置專案)。

步驟 6.3：創建代碼轉換項目

若要建立程式碼轉換專案

1. 登入 CodeBuild 主控台，然後選擇 [建立組建專案]。
2. 在「專案組態」區段中，提供專案的名稱 (例如`3-awsm2ccm-convert`)。
3. 在 [來源] 區段中，針對 [來源提供者] 選擇 Amazon S3，然後選擇您先前建立的輸入儲存貯體 (例如`codebuild-regionId-accountId-bucket`)。
4. 在 S3 物件金鑰或 S3 資料夾欄位中，輸入**prj_codebuild_01**。
5. 在「環境」區段中，選擇「自訂影像」。
6. 在「環境類型」欄位中，選擇「Linux」。
7. 在 [影像登錄] 下，選擇 [其他登錄]
8. 在 [外部登錄 URL] 欄位中，遵循下列the section called “AWS Mainframe Modernization 容器”章節。
9. 在 [服務角色] 底下，選擇 [現有服務角色]，然後在 [角色 ARN] 欄位中選擇您先前建立的服務角色；例如，`IAMRoleTaskExecutionRoleForCodeBuild`。
10. 展開 [其他組態] 區段，執行下列動作：

    a. VPC：根據您的設置進行配置（如果需要）。

    b. 逾時：設定為 60 分鐘。

    c. 佇列逾時：設定為 480 分鐘。

    d. 加密：選擇適當的加密設定 (預設值沒問題)。

    e. 在「環境變數」區段中，逐一新增下列項目：

- 名稱:PROJECT_ BUCKET. 值：**codebuild-regionId-accountId-bucket**。類型：明文
- 名稱:PROJECT_ DIR. 值：**prj_codebuild_01**。類型：明文
- 名稱:AWSM2CCM_ ACTION. 值：**conversion**。類型：明文
- 名稱：AWSM2CCM_ LOGGING _ BUCKET. 值：**s3:// codebuild-regionId-accountId-bucket**。類型：明文

11. 在 [Buildspec] 區段中，選擇 [插入建置命令]，然後選擇 [切換至編輯器]。
12. 用以下內容替換當前值：

```
version: 0.2
phases:
    build:
        commands:
            - export AWSM2CCM_PUSH_RUNTIME_COPYBOOKS=y
            - ln -s $CODEBUILD_SRC_DIR $PROJECT_DIR
            - . /app/awsm2ccm_prod/bin/setup_env.sh
            - run_awsm2ccm.sh $PROJECT_DIR
artifacts:
    files:
        - '*.log'
        - '_Converted/*/*'
        - '_Reports/*'
    discard-paths: no
    base-directory: $PROJECT_DIR
```

其中，PROJECT_ DIR 是可用的環境變量 CodeBuild。如需詳細資訊，請參閱建置環境中的環境變數。

13. 在「成品」區段中，執行下列動作：

- 在 [類型] 下，選擇 Amazon S3，然後選擇您的輸出儲存貯體 (例如codebuild-regionId-accountId-bucket)。
- 輸入做為「路徑」ARTIFACTS。
- 對於「名稱」，輸入**prj_codebuild_01**。
- 對於 Artifact 封裝，選取無。
- 針對覆寫人工因素名稱，取消勾選此選項。
- 對於加密，請將其保留為默認設置。

14. 針對「記錄檔」區段，執行下列動作：

    - CloudWatch 記錄檔：已停用
    - S3 日誌：已啟用

    - 鏟斗:**codebuild-regionId-account-bucket**
    - 記錄檔路徑：**CODEBUILD-LOGS**

15. 選擇 Create build project (建立建置專案)。

第 7 步：定義項目並上傳源代碼

「定義專案」會設定專案資料夾和組態檔案，並以預設組態進行初始化。在此步驟中，您將啟動組建。若要執行此作業：

1. 登入 AWS CodeBuild 主控台。
2. 在左側導覽窗格中，選擇 [建置專案]。
3. 選擇先前創建的項目（`1-awsm2ccm-define-project`）進行構建
4. 選擇 [開始建置]，然後選擇 [立即開始] 以定義專案。一旦建置開始，狀態就會變更為 [進行中]。
5. 選擇「階段詳細資料」以查看由 AWS CodeBuild 專案編排的每個步驟的進度。
6. 請等待所有步驟的狀態變更為「成功」。
7. 轉到 Amazon S3 控制台。
8. 找到並點擊 Amazon S3 存儲桶命名 `codebuild-regionId-accountId-bucket`

    - **CODEBUILD-LOGS/**文件夾包含正在運行的 AWS CodeBuild 項目的 AWS CodeBuild 日誌。
    - **prj_codebuild_01/**包含專案結構的資料夾。它在分析，擴展宏和轉換步驟中使用。您可以選擇`prj_codebuild_01/`探索細節
    - **cobol_reserved.rsw**為轉換器保留的配置文件（單COBOL詞列表）。它在轉換步驟中使用。
    - **Macro_Expansion/**文件夾包含宏擴展到彙編程序。它在展開宏步驟中使用。
    - **macro_settings.json**組態檔案包含自訂巨集取代。它在展開宏步驟中使用。
    - **macrolib/**資料夾包含要轉換的彙編程式巨集。它在分析和轉換步驟中使用。

        1. 選取 `macrolib/`。

2. 依預設，會提供一個名MACRO1.mac為的彙編程式巨集做為範例檔案。刪除此檔案，因為分析不需要此檔案。
3. 將您的宏上傳到此目錄中。

- **project_settings_aux.json**配置文件包含與字碼頁相關的設置。它在轉換步驟中使用。
- **project_settings.json**組態檔案包含轉換器的設定。它在轉換步驟中使用。
- **srclib/**文件夾包含要轉換的彙編程序。它在分析和轉換步驟中使用。

1. 選擇 srclib/。
2. 默認情況下，兩個名為SQtest01.asm和SQtest02.asm的彙編程序作為示例提供。刪除這些文件，因為它們不需要您的分析和轉換。
3. 在此目錄中上傳您的彙編程序。

9. 驗證1-awsm2ccm-define-project步驟的狀態。它應該已經在最新的構建狀態選項卡下成功。

您已準備好進行下一步：程式碼分析。

步驟 8：執行分析並瞭解報告

> **Note**
>
> AWS Mainframe Modernization 代碼轉換分析步驟是免費的。

在此步驟中，您將啟動另一個組建：

1. 在左側導覽窗格中，選擇 [建置專案]。
2. 選擇您在步驟 6.2 中創建的項目以構建：2-awsm2ccm-analysis。
3. 選擇 [開始建置]，然後選擇 [立即開始] 以產生分析報告。這將啟動構建並將狀態更改為正在進行中。
4. 選擇階段詳細資料，您可以在其中看到 AWS CodeBuild 專案精心策劃的每個步驟的進度。請等待所有步驟的狀態變更為「成功」。
5. 從 AWS Management Console，移至 Amazon S3 服務主控台。
6. 找到並單擊 Amazon S3 存儲桶：codebuild-regionId-accountId-bucket

   a. **ARTIFACTS/**資料夾包含分析和轉換步驟的輸出。

   b. 選擇 ARTIFACTS/prj_codebuild_01/_Reports/。

c. 下列報告可供使用：

- AWSM2CCM-Analysis-Report-<timestamp>.pdf是一份執行報告，提供 AWS Mainframe Modernization 程式碼轉換計費和範圍，改善轉換、轉換摘要，以及詳細的轉換統計資料。它也會彙總專案層級的程式碼計數和可計費程式碼計數，並提供每個元件的量度和參照成員清單。在執行實際轉換之前，執行和檢查此報表非常重要。
- Conversion_Detailed_Statistics.txt為每個組件中找到的每個指令提供頻率和預期的轉換結果（顯示為「轉換狀態」）。這提供了一種快速的方法來識別指令是否清楚表明轉換器不支持。可能的轉換狀態結果為：
  - 完全轉換：指令將被準確轉換為COBOL。
  - 部分轉換：支援指令，但使用不支援的參數或運算式。轉換後可能需要手動調整。
  - 未轉換：轉換器不支持該指令。
  - 預編譯指令以驗證：這些通常包含在宏中，並參考大型機上可能也被稱為條件彙編語言（例如AIF，AGO）指令。這些由這些指令或指令驅動的預編譯器處理，可以選擇並生成乾淨/ASM靜態代碼。這些指令取決於被編譯的 Macro 參數的實際值。因此，相同的宏可以生成不同的ASM代碼片段，具體取決於傳遞的參數的值。這是因為存在這樣的預編譯指令。在這種情況下，請考慮擴展或重新設計宏。
- Conversion_Global_Statistics.txt提供元件層級的轉換狀態摘要。
- CrossReference_PgmToCpyMacro.txt彙編程序程序對宏的依賴關係的報告。它提供了一種快速的方法來確定上傳的代碼中是否缺少任何宏。
- CrossReference_PgmToPgm.txt報告彙編程序對其他彙編程序的依賴關係。它提供了一種快速的方法來確定是否有任何彙編程序從上傳的代碼丟失。

7. 返回 AWS CodeBuild 服務主控台。
8. 驗證兩個 awsm2cc 分析步驟的狀態。它應該已經在最新的構建狀態選項卡下成功。

您已準備好進行下一步：代碼轉換。

步驟 9：執行程式碼轉換

> **Important**
>
> AWS Mainframe Modernization 代碼轉換步驟將根據您的用量計費。如需計費的詳細資訊，請參閱the section called “瞭解程式碼轉換計費”。

在此步驟中，您將配置轉換過程，然後啟動構建。

1. 從 AWS Management Console，轉到 Amazon S3 服務。
2. 找到並點擊 Amazon S3 存儲桶：codebuild-regionId-accountId-bucket。
   a. 前往 prj_codebuild_01/。
   b. 選取project_settings.json，然後選擇 [下載]。
   c. 開啟project_settings.json檔案以查看下列JSON結構：

```
{
"Source programs directory":"srclib",
"Source copybooks/macros directory":"macrolib",
"Copybook/Macros Conversion":"Called_only",
"Do not regenerate the Copy/Macro if already exists":"false",
"Target Compiler":"IBM",
"Endianess":"Big",
"Converted programs extension":"",
"Converted CICS programs extension":"",
"Converted copies/macros extension":"",
"Trace Level":"STANDARD",
"Trace file open mode":"append",
"Data definition level":5,
"Start picture column":40,
"Generate Sync FILLER with name":"FILL-SYNC",
"Use SYNC clause":"yes",
"Decimal Point Comma":"true",
"Original Source Placement":"RIGHT"
}
```

   其中，

   - 來源程式目錄：包含轉換所需的彙編程式。
   - 來源文獻/巨集目錄：包含轉換所需的彙編程式巨集和文字本。
   - 文案/宏轉換可以是：
     - 全部：此單選按鈕表示完整轉換將轉換目錄中所有可用的 Copybook /宏，而不考慮程序是否正在使用。
     - 呼叫 _only：此選項按鈕表示完整轉換只會轉換程式實際使用的 Copybook /巨集。

- > **Important**
  >
  > 如果複製/巨集已存在，則不需要重新產生該複製/巨集。

  當這是真實的時候，如果它已經轉換（存在於輸出文件夾中），該工具將不會再次轉換Copybook /宏。
- 目標：程序（生成的代碼）的轉換取決於目標COBOL編譯器。支援下列選項：
  - 「IBM" 表示IBM大型主機
  - 微型對焦功能的「MF」COBOL
  - 對於維揚特而言，「VERYANT」是 COBOL
  - NTTDATA企業版的「NTT」COBOL (優尼基克斯)
- 位元和位元：程式 (產生的程式碼) 的轉換取決於目標平台 (位元/位元組)。此組合允許選取下列支援的選項：
  - 位序：大（對於大端）/小（小端）。例如，IBMz/OS 大型主機是大端，視窗是小端序，Linux 因分佈而異（例如，Amazon Linux 2 上是小端）。EC2
  - 位：32/64（如果沒有給出，默認值將是 32）。建議的設定為 32 位元。
- 轉換後的程序擴展名：這是為生成的COBOL程序設置文件擴展名。空白（「」): 無副檔名。對於 Micro Focus COBOL 目標，建議使 Micro Focus 企業開發人員能夠正確辨識CBL這些檔案。
- 轉換後的程CICS序擴展名：這是為生成的CICSCOBOL程序設置文件擴展名。空白（「」): 無副檔名。對於 Micro Focus COBOL 目標，建議使 Micro Focus 企業開發人員能夠正確辨識CBL這些檔案。
- 轉換的版本/宏擴展名：這是為生成的字帖設置文件擴展名。COBOL空白（「」): 無副檔名。對於 Micro Focus COBOL 目標，建議使 Micro Focus 企業開發人員能夠正確辨識CPY這些檔案。
- 追蹤層級：追蹤是在轉換 CodeBuild 期間使用記錄的資訊。用戶可以通過選擇提供的選項中的任何一個選項來選擇詳細級別。
  - ERROR= TRACEERROR：僅顯示轉換錯誤。
  - STANDARD= TRACESTANDARD：顯示轉換錯誤和標準信息。這是建議的設定。
  - ALL= TRACEALL：追蹤的最大層級
- 追蹤檔案開啟模式：未使用。建議使用附加的預設設定。

- 數據定義級別：這表示在工作-存儲和鏈接部分中定義的子字段的初始級別（級別「01」之後）。必須是數字。
- 開始圖片列：這是關於生成的COBOL代碼的格式，並指示PIC子句放置的列（在字段名稱之後）。必須是數字。
- 原始源位置：這表示註釋放置在程序中的位置。它有兩個選項：
  - RIGHT：此選項會將評論或其他信息放在第七十三（73）列之後的正確位置。在代碼COBOL中寫在前七十二（1-72）列中，第七十三列（> = 73）列中的任何內容都將被視為註釋。
  - ABOVE：此選項會將評論放置在翻譯內容的上方。
- FILLER使用名稱生成同步：此選項與二進制字段（彙編程序「H」，「F」，「D」數據類型，轉換為「」數據類型）的內存中的對齊有關。COBOL COMP為了保證正確的對齊邊界，將在轉換過程中添加明確填充字段。這是一個基於文本的選項，值必須是一個字符串（如 FILL-SYNC）。
- use SYNC 子句：此選項是指二進制字段內存中的對齊方式。是 = 轉換為的所有欄位COBOL。「COMP」將以條款「SYNC」定義 (例如 05 WRKFLD PIC S9 (09) COMPSYNC)。
- 小數點逗號：當這是真的，DECIMAL-POINT IS COMMA 子句將被添加到「SPECIAL-NAMES」COBOL 段落。

d. 根據您的需求，變更適當的參數，然後儲存`project_settings.json`.

e. 從 Amazon S3 儲存貯體`prj_codebuild_01/`中移除現有`project_settings.json`檔案，然後上傳新版本。

3. 返回到服 AWS CodeBuild 務。
4. 選取您先前建立的要建置的專案：`3-awsm2ccm-convert`

a. 選擇 [開始建置]，然後選取 [立即開始]，將彙編程式和巨集轉換為COBOL程式和撰寫本。

b. 等待此專案的建置狀態變更為 [成功]。它將位於「最新版本狀態」選項卡下。

步驟 10：驗證代碼轉換

1. 從 AWS Management Console，轉到 Amazon S3 服務。
2. 找到並點擊 Amazon S3 存儲桶：`codebuild-regionId-accountId-bucket`。
3. 導覽至**`awsm2ccm-do-not-delete`** 。 AWS Mainframe Modernization 在轉換過程中，代碼轉換會為每個彙編程序或宏模塊創建編碼的二進制文件。這些文件是防止重複計費給客戶必不可少，

也跟踪多少提供彙編代碼進行了分析和轉換。檔案儲存在下列位置：codebuild-regionId-accountId- bucket/awsm2ccm-do-not-delete/<your_AWS_account_id>/Hash。編碼的文件不包含任何彙編代碼，也不可能從這些文件中提取客戶代碼。

> **Important**
>
> 既不手動編輯這些文件或刪除這些文件。編輯或刪除這些檔案可能會導致相同元件產生多個帳單。

將**awsm2ccm-do-not-delete/**資料夾視為系統管理的目錄。對此目錄或其內容進行任何變更 AWS Support 之前，請先諮詢。

4. 按codebuild-regionId-accountId-bucket一下以返回值區。
5. 選擇**ARTIFACTS/prj_codebuild_01/**。 轉換/資料夾包含作為程式碼轉換步驟的結果所產生的COBOL輸出。它將具有以下子目錄：
   - 字帖/文件夾包含生成COBOL的字帖。
   - 程序/文件夾包含生成的COBOL程序。
   - runtime_lib/ 文件夾包含由解決方案提供的附加COBOL程序和字帖。
6. 如果「分析報告」和其他報告指出轉換成功，且 AWS CodeBuild 專案標記3-awsm2ccm-convert為「成功」，請從 _Conver ted/ 目錄下載COBOL程式碼和撰寫本。

步驟 11：下載轉換後的代碼

在此步驟中，從 _Converted/ 目錄下載COBOL代碼和文案，並在目標環境中編譯它們。COBOL

1. 從 AWS Management Console，轉到 Amazon S3 服務。
2. 找到並點擊 Amazon S3 存儲桶：codebuild-regionId-accountId-bucket。
3. 導航到位置： ARTIFACTS/prj_codebuild_01/_Converted/。
4. 從「_ 轉換」/下的所有子目錄下載轉換後的COBOL代碼。您也可以使用以下CLI命令一次下載它們：

```
aws s3 cp s3://codebuild-regionId-accountId-
bucket/ARTIFACTS/prj_codebuild_01/_Converted/ . --recursive
```

5. 在目標COBOL環境COBOL中分析並編譯轉換的內容。

清除資源

如果您不再需要為此教學課程建立的資源，請刪除這些資源以免產生額外費用。若要這樣做，請完成下列步驟：

- 刪除您為本教學課程建立的 S3 儲存貯體。如需詳細資訊，請參閱 Amazon 簡單儲存服務使用者指南中的刪除儲存貯體。
- 刪除您為此教學課程建立的IAM策略。如需詳細資訊，請參閱IAM使用指南中的刪除IAM策略。
- 刪除您為此教學課程建立的IAM角色。如需詳細資訊，請參閱IAM使用指南中的刪除角色或執行個體設定檔。
- 刪除您為此自學課程建立的 CodeBuild 專案。如需詳細資訊，請參閱AWS CodeBuild 使用指南 CodeBuild中的刪除組建專案。

查龍集成

介紹查龍 SSP

1987 年，太陽電腦公司發布了 SPARC V7 處理器，一個 32 位RISC處理器。SPARCV8 隨後於 1990 年-原始 SPARC V7 的修訂版，最顯著的包含硬件除法和乘法指令。SPARCV8 處理器形成了許多服務器和工作站（例如 SPARCstation 5，10 和 20）的基礎。一九九三年，SPARCV8 隨後是 64 位元 SPARC V9 處理器。這也成為了許多服務器和工作站的基礎，例如企業 250 和 450。

由於硬體過時，缺乏備用或翻新零件，為這些舊SPARC型工作站和伺服器開發的軟體和系統變得更難以維護。為了滿足對某些 end-of-life SPARC基於系統的持續需求，Stromasys S.A. 開發了模擬器產品的 Charon-系SSP列。SPARC下列產品是針對指定原生硬SPARC體系統的軟體式虛擬機器取代品。以下是模擬硬體系列的一般概述。

夏龍-SSP /4M 模擬以下硬件：SPARC

- Sun 4m 系列（由 Sun SPARCstation 20 表示）：最初，基於 600MP 系列中引入的處理器模塊總線的多MBus處理器 Sun-4 變體。SPARCServerSun 4m 架構後來還包含了非MBus單處理器系統，例如 SPAR Cstation 5，利用 V8 架構處理器。SPARC從 SunOS 4.1.2 開始支援，並由索拉瑞斯 2.1 到索拉瑞斯 9 支援。SPARCServer在索拉里斯 2.5.1 之後，600 萬像素的支撐被下降。

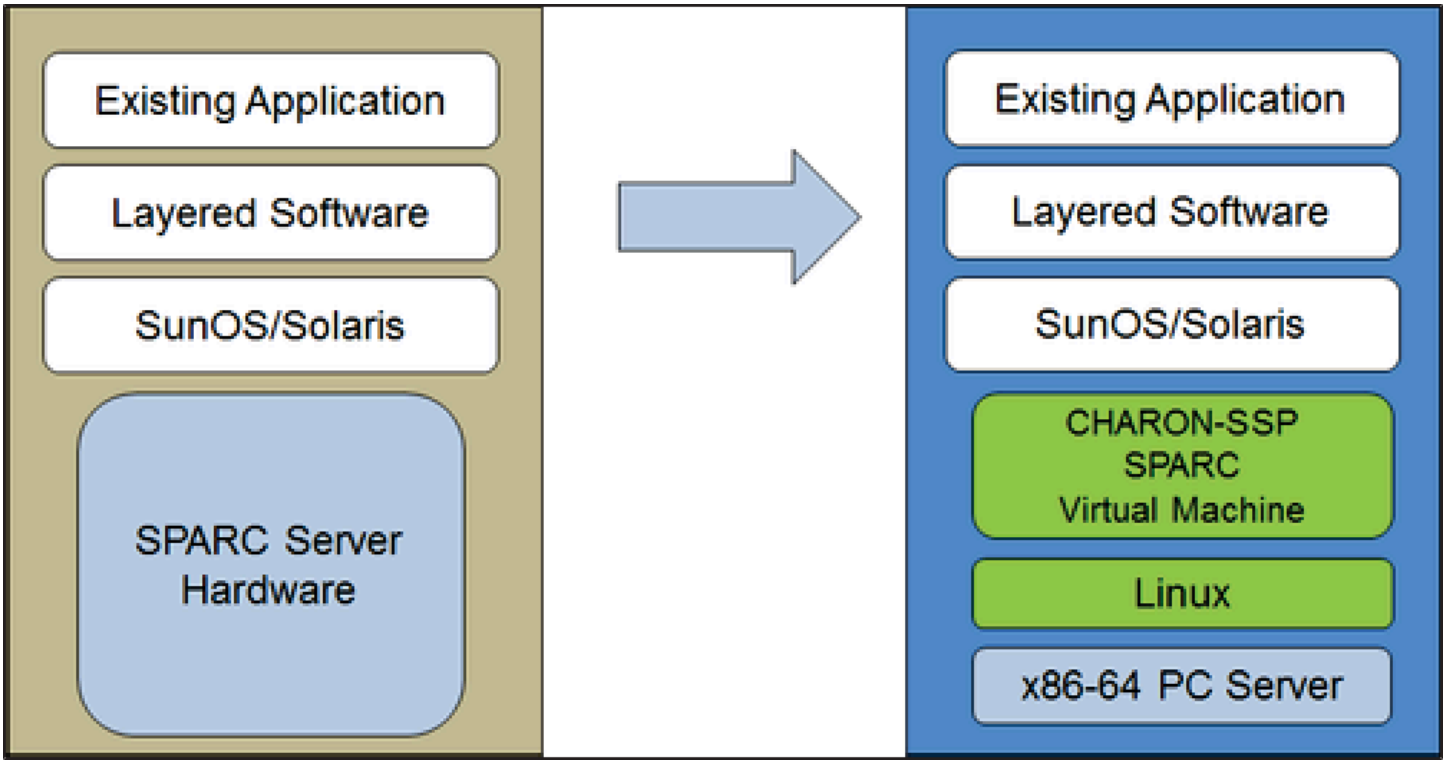
夏龍-SSP /4U (+) 會模擬下列硬體：SPARC

- Sun 4u 系列（由 Sun 企業 450 代表）：（U 為超SPARC）-這種變體引入了 Sun 超系列中首次使用的 64 位 SPARC V9 UPA 處理器架構和處理器互連。從 2.5.1 版開始，支援 32 位元版本的 Solaris。Sun 4u 的第一個 64 位元 Solaris 發行版本是 Solaris 7。超SPARC我的支持在索拉里斯 9 之後被放棄。Solaris 10 支援從超音波ARC第二到超 IV 的 Sun 4u 實作。SPARC

夏龍-SSP /4V (+) 會模擬下列硬體：SPARC

- Sun 4V 系列（由 SPARC T2 和 T4 表示）：這種變化增加了虛擬化管理程序處理器虛擬化到 Sun 4u; 在超 T1 多核處理器中引入。SPARC從 3/05 版開始，Solaris 第 10 版支援選取的硬體 HW2 (大多數型號-包括由 Charon-SSP 模擬的硬體) 都需要較新版本的 Solaris 10)。還支援數個 Solaris 11 版本。

下圖顯示了將物理硬件遷移到仿真器的基本概念。



Charon-SSP 虛擬機器可讓 Sun 和 Oracle 電腦SPARC的使用者以不需要變更原始系統組態的方式，取代其原生硬體。這表示您可以繼續執行應用程式和資料，而不需要切換或移植到其他平台。Charon-SSP 軟體在商品上執行，Intel 64 位元系統可確保持續保護您的投資。

夏龍-SSP /4U+ 支援與夏隆-/4 SSP U 相同的虛擬SPARC平台，而夏隆-/4V+ 與夏隆-/4V 相同。SSP SSP然而，4U+ 和 4V 以上版本利用 Intel 的/EPT和 AMD AMD-vVTx/NPT硬體輔助虛擬化技術，提供更好的虛擬化效能。CPUs CPUCharon-SSP /4U+ 和夏朗-/ SSP 4V+ 需要支援 VT-x/ EPT 或 AMD-NPT v/，且必須CPUs安裝在專屬主機系統上。不支援在虛擬機器中執行這些產品變體 (例如，在 VMware)。

> **Note**
>
> 如果您打算在雲端環境中執行 Charon-SSP /4U+ 或 4V+，請聯絡 Stromasys 或 Stromasys 討論您的需求。VAR

支援的客體作業系統

Charon-SSP /4M 虛擬機器支援下列客體作業系統版本：

- 蘇諾斯 4.1.3-4.1.4

- 索拉里斯 2.3 至索拉里斯 9

Charon-SSP /4U (+) 虛擬機器支援下列客體作業系統版本：

- 索拉里斯 2.5.1 至索拉里斯 10

Charon-SSP /4V (+) 虛擬機器支援下列客體作業系統版本：

- 索拉里斯 10 (從更新 4, 08/07 開始) 和索拉里斯 11.1 至索拉里斯 11.4

對於夏龍-SSP /4V (+)，請注意以下事項：

- 對於模擬的 SPARC T4，支援的索拉瑞斯 10 版本有：甲骨文索拉瑞斯 10 1/13、甲骨文索拉瑞斯 10 8/11 和索拉里斯 10 9/10，或含甲骨文索拉瑞斯 10 8/11 修補程式集的索拉里斯 10 10/09。
- 模擬的 SPARC T4 模型是在模擬器中執行 Solaris 11.4 的先決條件。
- 不支援 Solaris 核心區域。

Charon SSP 雲端執行個體先決條件

透過選取執行個體類型或形狀，您可以選取將用於雲端中 Charon-SSP 主機執行個體的虛擬硬體。因此，選取執行個體類型或形狀會決定 Charon-SSP 虛擬主機硬體的硬體特性 (例如，您的虛擬 Charon 主機系統將擁有多少CPU核心和記憶體數量)。

> **Note**
>
> 如果您使用 Charon-SSP 市集映像檔來啟動執行個體，則可滿足所有 Linux 主機作業系統需求。

最低硬體需求說明如下。

關於尺寸指南的要點:

- 特別是關於主機CPU核心和主機記憶體數目的大小調整準則-顯示最低需求。必須檢閱每個部署情況，並視需要調整實際的主機大小。例如，如果客體應用程式產生高 I/O 負載，則必須增加 I/O 可用的CPU核心數目。此外，具有許多模擬的系統通CPUs常能夠產生更高的 I/O 負載，因此可用於 I/O 的CPU核心數量可能需要增加。在超執行緒環境中，為了獲得最佳效能，CPU核心數 (即實際/

實體CPUs) 必須足以滿足使用中模擬器的CPU需求，因此避免高工作負載執行緒共用一個實體核心。CPU

- I/O 處理的模擬CPU核心CPUs和CPU核心配置由組態決定。請參閱一般 Charon-SSP 使用者指南中的CPU組態，以取得有關此項目的詳細資訊，以及 I/O 處理的預設CPU核心配置。

> ⚠ 重要的一般資訊
>
> - 為了方便將模擬器數據從一個雲實例快速傳輸到另一個雲實例，強烈建議將所有相關的模擬器數據存儲在單獨的磁盤卷上，該磁盤卷可以輕鬆地從舊實例分離並附加到新實例。
> - 確保從頭開始正確地標註您的例證尺寸（請查看下面的最低需求）。Charon-SSP AL 的 Charon-SSP 授權會在執行個體首次啟動時建立。稍後變更為其他執行個體大小/類型，從而變更CPU核心數量會使授權失效，因此無法啟動 Charon 執行個體 (需要新的執行個體)。如果計劃在 AutoVe 模式下使用 Charon-SSP AL 實例，請務必在首次啟動之前包括 AutoVe 伺服器資訊，否則將使用公用授權伺服器。Charon-SSP VE 的許可證是根據許可證服務器上獲取的指紋創建的。如果授權伺服器直接在模擬器主機上執行，而模擬器主機稍後需要變更CPU核心數目，則授權將失效 (可能需要新的授權，也可能需要新的執行個體)。

實例先決條

一般CPU要求：Charon-SSP 支援以現代 x86-64 架構處理器為基礎的 Amazon 執行個體。EC2

夏龍的最低要求：SSP

- 主機系統CPU核心數目下限：
  - 主機作業系統至少有一個CPU核心，再加上：
  - 對於每個模擬SPARC系統：
    - 每個執行個體模擬CPU的一個CPU核心，再加上：
    - 至少一個額外的 I/O 處理CPU核心 (至少兩個核心，如果使用伺服器JIT最佳化)。有關CPU配置選項，請參閱上面提到的配置部分。默認情況下，Charon 會將 Charon 主機CPUs可見數的 1/3（最小 1; 四捨五入）分配給 I/O 處理。
- 最低記憶體需求：
  - 對於 Linux 主機作業系統而言，RAM需要 4GB 或更大的空間。實際需求可能會更高，並且取決於 Linux 主機上運行的非模擬器服務的需求。之前針對 Linux 主機至少 2GB 的建議仍然RAM適用於許多系統，但是 Linux 作業系統和應用程式的需求不斷增加，導致對新安裝的更新建議。加：

  - 對於每個模擬SPARC系統：
    - 模擬執行個體的設定記憶體，以及：
    - 2GB 的 RAM (RAM如果使用伺服器JIT則為 6GB)，以允許DIT最佳化、模擬器需求、執行階段緩衝區SMP和圖形模擬。
- 如果在現代 x86-64 上啟用了超執行緒CPUs，則兩個執行緒可以在一個實體CPU核心上執行，為主機作業系統提供兩個邏輯CPUs。如果可能，請停用 Charon-SSP 主機上的超執行緒。但是，在VMware和雲環境中通常不可能這樣做，或者尚不清楚是否使用超線程。Charon-SSP 超執行緒選項可讓 Charon-SSP 適應此類環境。有關詳細的CPU配置信息，請參閱上面提到的一般 Charon-SSP 用戶指南中的配置部分。P 租賃注意事項：為了獲得最佳性能，Charon-SSP 線程不應共享物理內CPU核-主機系統上應該有足夠的物理內核以滿足配置的仿真器的要求。
- 一個或多個網絡接口，具體取決於客戶的要求。
- Charon-SSP /4U+ 和 Charon-/SSP4V+ 必須在支援 Intel VT-x/ EPT 或 AMD-v/ NPT (裸機執行個體) 的實體硬體上執行，因此無法在所有雲端環境中執行。請查看雲端供應商的說明文件，瞭解此類硬體的可用性。此外，請注意以下幾點：
  - 只有在使用斯特羅馬斯支援的 Linux 核心時，才能使用 Charon-SSP SSP /4U+ 和 Charon-/4V+。
  - 如果您需要這種類型的模擬SPARC硬件，請聯繫 Stromasys 或您的 Stromasys VAR 以詳細討論您的需求。

為 Charon 建立和設定 AWS 雲端執行個體 (新增GUI)

本節反映了 2022 AWS Management Console 年春季的情況。如果您仍然使用舊版主機，請參閱 Charon-SSP AWS 入門指南的附錄。

一般先決條件

此說明顯示中 Linux 執行個體的基本設定 AWS。它不會列出特定的必要條件。但是，根據您的使用案例，請考慮下列先決條件：

- Amazon 帳戶和 AWS Marketplace 訂閱
  - 若要在中設定 Linux 執行個體 AWS，您需要具有管理員存取權的 AWS 帳戶。
  - 識別您計劃在其中啟動執行個體的 AWS 區域。確保您計劃使用的 AWS 服務可在該地區使用。請參閱AWS 服務 (按地區)。
  - 識別您計劃在其中啟動執行個體的子網路VPC和子網路。

- 如果您的執行個體需要網際網路存取，請確定與您相關聯的路由表VPC具有網際網路閘道。如果您的執行個體需要內部部署網路的VPN存取權，請確定VPN閘道可供使用。您VPC及其子網路的確切組態將取決於您的網路設計和應用程式需求。
- 若要訂閱特定 AWS Marketplace 服務，請在中選擇 [AWSMarketplace 訂閱]， AWS Management Console 然後選擇 [管理訂閱]。
- 搜尋您計劃使用的服務並訂閱。成功訂閱後，您將在「管理訂閱」部分中找到訂閱。從那裡你可以直接啟動一個新的實例。

- 執行個體硬體和軟體先決條件會因執行個體的規劃使用方式而有所不同：
  - 選項 1：將實例用作 Charon 仿真器主機系統：
    - 請參閱 Charon 產品的《使用者指南》和/或《入門指南》中的硬體和軟體先決條件章節，以確定 Linux 執行個體必須符合的確切硬體和軟體先決條件。您用來啟動執行個體的映像檔以及您選擇的執行個體類型決定了雲端執行個體的軟體和硬體。
    - 執行模擬的舊系統需要 Charon 產品授權。請參閱 Charon 產品文件中的授權資訊，或聯絡您的 Stromasys 代表或 Stromasys 以取得其他資訊。VAR
  - 選項 2：執行個體將用作專用 VE 授權伺服器：
    - 如需詳細必要條件，請參閱《VE 授權伺服器指南
- 某些可以在 Charon 仿真器產品提供的模擬系統中運行的舊版操作系統需要操作系統的原始供應商的許可證。用戶負責與舊版操作系統相關的任何許可義務，並且必須提供適當的許可證。

使用啟 AWS Management Console 動新執行個體

若要建立新執行個體

1. 登錄 AWS Management Console 並在打開 Amazon EC2 控制台https://console.aws.amazon.com/ec2/。
2. 選擇啟動執行個體。
3. 輸入執行個體的名稱。
4. 選取一個AMI。AMI是用於啟動雲端執行個體的預先封裝映像檔。它包括操作系統和適用的應用軟件。的選擇AMI取決於您計劃使用執行個體的方式：

   - 如果將實例用作 Charon 仿真器主機系統，則可以選擇以下幾種AMI選擇：
     - 從預先封裝的 Charon 市集映像檔安裝 Charon 主機系統：它們包含基礎作業系統和預先安裝的 Charon 軟體。
       - 請洽詢您的 Stromasys 代表您的雲端供應商市場目前有哪些選項可供使用。

    - 根據雲提供商和 Stromasys 產品發布計劃的不同，可以有兩種變體：
        - 自動授權 (AL)，可與由 Stromasys 操作的公用授權伺服器搭配使用，或與私人、客戶操作的 AutoVE 授權伺服器搭配使用
        - 虛擬環境 (VE)，可搭配私人、客戶操作的 VE 授權伺服器使用
    - 使用傳統的 Charon 模擬器安裝搭配 Linux 的 Charon 模擬器安裝RPM套件來安裝 Charon 主機系統：
        - 選擇您所選 Charon 產品和版本支援的發行版本AMI的 Linux。請參閱 Stromasys 文件網站上的產品使用者指南。
    - 如果要將執行個體用作專用 VE 授權伺服器，請參閱授權文件中的《VE 授權伺服器指南》，瞭解 Linux 執行個體的需求。

    在您決定需要AMI哪一項之後，請選取符合的 Linux 或 Charon 產品AMI。如果沒有看到您需要AMI的項目，請選擇 [瀏覽更多] AMIs。選擇與您計劃使用執行個體的方式相符的 Linux AMI。可為下列其中之一：

    - 預先包裝的 Charon VE 市場形象。的名稱AMI將包含字串「ve」。
    - 用於自動授權或 AutoVe 的預先包裝的 Charon AL 市場映像。
    - 支援RPM產品安裝的 Linux 版本。
    - VE 授權伺服器支援的 Linux 版本。

5. 選取執行個體類型。Amazon EC2 提供具有不同組合CPU、記憶體、儲存和聯網容量的執行個體類型。選取符合您要使用之 Charon 產品需求的執行個體類型。某些市集映像檔的執行個體類型選擇受到限制。
6. 選取現有 key pair，或建立並儲存新金鑰配對。如果您選取現有的 key pair，請確定您擁有相符的私密金鑰。否則，您將無法連線至執行個體。

    > **Note**
    >
    > 如果您的管理系統支援此功能，對於 RHEL 9.x、岩石 Linux 9.x 和 Oracle 9.x 版，請使用SSH金鑰類型或. ECDSA ED25519 這些類型允許您使用通SSH道連接到這些 Charon 主機 Linux 系統，而無需將 Charon 主機上的默認加密原則設置更改為較不安全的設置。例如，這對於 Charon-SSP 經理很重要。請參閱 Red Hat 說明文件中的使用全系統密碼編譯原則。

7. 在 [網路設定] 區段中，選擇 [編輯]。選擇與您的環境相對應的設定。

- 指定一個VPC。
- 指定現有的子網路或建立新子網路。
- 啟用或停用自動將公用 IP 位址指派給主要介面。只有在執行個體具有單一網路介面時，才能自動指派。
- 指派現有或新的自訂安全性群組。安全性群組必須SSH至少允許存取執行個體。您計劃在執行個體上執行的應用程式所需的任何連接埠也必須允許。您可以在建立執行個體之後隨時修改安全性群組。

8. 在儲存區段中，針對根磁碟區 (系統磁碟)，選擇適合您環境的大小。對於 Linux 系統，建議的最小系統磁碟大小為 30 GiB。若要為虛擬磁碟容器和其他儲存區需求提供空間，您可以立即或在啟動執行個體之後新增更多儲存空間。但是系統磁碟大小必須涵蓋 Linux 系統需求，包括您計劃安裝的任何應用程式和公用程式。

> **Note**
>
> 我們建議您為 Charon 應用程式資料 (例如磁碟映像檔) 建立個別的儲存磁碟區。如有需要，您可以稍後將此類磁碟區移轉到另一個執行個體。

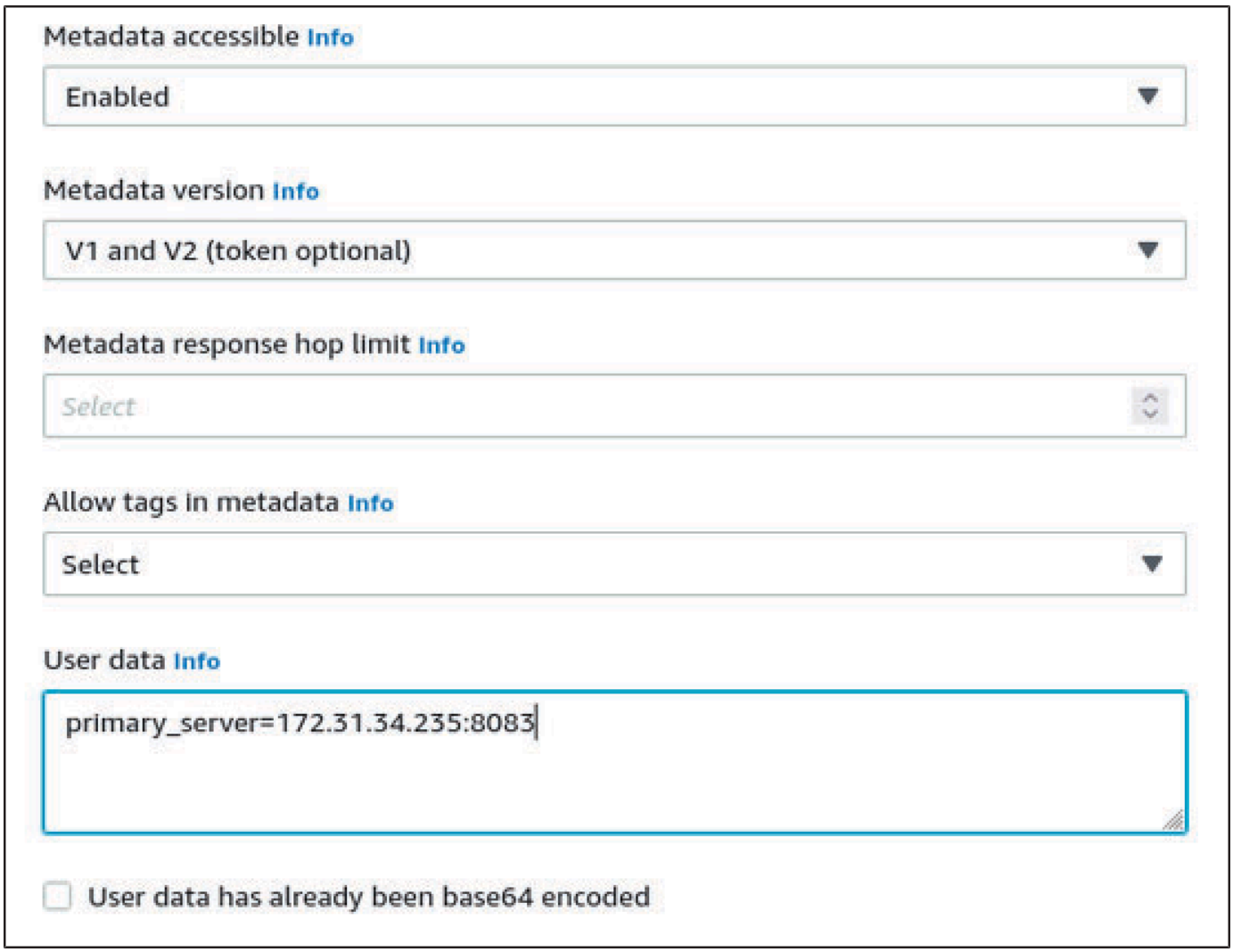
9. 展開 [進階詳細資料] 區段，向下捲動，然後選取 [指定CPU選項]。下面的圖像中顯示了三個更可能對 Charon 仿真器環境有用的示例。

10. 對於版本早於 1.1.23 的 VE 授權伺服器系統，您必須將必要的IAM角色指派給執行個體。它必須是允許`ListUsers`動作的角色。若要指派角色，請在展開的 [進階詳細資料] 區段中選取 [IAM執行

個體設定檔] 下的角色，或選擇 [建立新IAM設定檔]。如需詳細資訊，請參閱 Amazon 的IAM角色 EC2。

11. 如果您的執行個體是以 Charon AL AWS Marketplace 映像檔為基礎，而且您打算使用 Stromasys 操作的公用授權伺服器，則在啟動執行個體之前，必須將對應的資訊新增至執行個體組態。

    輸入 AutoVe 授權伺服器的資訊，如以下影像所示。



    以下是有效的使用者資料組態選項：

    - **primary_server=**<*ip-address*>[**:**<*port*>]
    - **backup_server=**<*ip-address*>[**:**<*port*>]

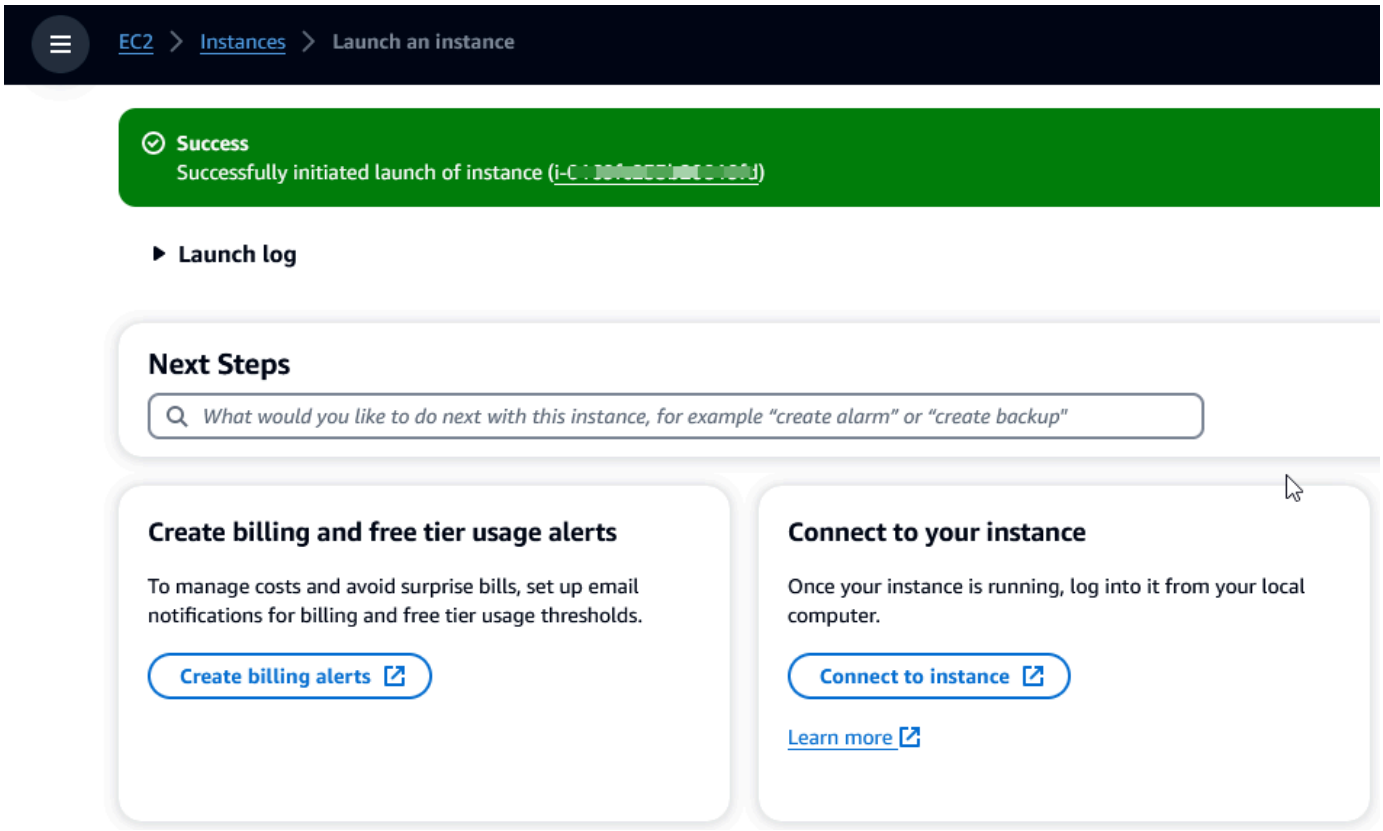
    位置

    - <ip-address>代表主伺服器和備份伺服器的 IP 位址 (如適用)。

- <port>代表用於與許可證伺服器通訊的非預設TCP連接埠（預設值：TCP/8083）。

> **Note**
>
> 在初始啟動時，至少必須設定一個授權伺服器才能啟用 AutoVe 模式。此執行個體將會連結至由 Stromasys 操作的其中一個公用授權伺服器。

12. 在 [摘要] 區段中，選擇 [啟動例項]。一段時間後，您將看到以下成功消息：



13. 在畫面右下角，選擇 [檢視所有實體]。
14. 若要查看執行處理的詳細資訊，請選取「執行處理」表格中代表執行處理之列左側的核取方塊。您的執行個體詳細資訊會顯示在畫面下半部。有關如何連接到執行個體的詳細資訊，請參閱 Amazon EC2 使用者指南中的 Connect。

AWS 大型主機現代化重新平台 NTT DATA

AWS 大型主機現代化提供各種 Amazon 機器映像 ()。AMIs這有AMIs助於快速佈建 Amazon EC2 執行個體，並透過使用資料建立量身打造的環境，以重新託管和重新平台中的大型主機應用程式。 AWS NTT本指南提供存取和使用這些步驟所需的步驟AMIs。

必要條件

- 確保您擁有可以建立 Amazon EC2 執行個體的 AWS 帳戶的管理員存取權。
- 確認您計劃建立 Amazon 執行個體的區域中是否提供 AWS 大型主機現代化服務。EC2請參閱按地區提供的AWS服務清單。
- 確定您要VPC在哪裡創建 Amazon EC2 實例的 Amazon。

訂閱 Amazon 機器映像

當您訂閱 AWS Marketplace 產品時，您可以從產品的執行個體啟動執行個體AMI。

1. 登入 AWS Management Console 並在 https://console.aws.amazon.com/Marketplace 開啟 AWS Marketplace 主控台。
2. 選擇 Manage subscriptions (管理訂閱)。
3. 將以下連結複製並貼到瀏覽器的網址列 https://aws.amazon.com/marketplace/ ：
4. 選擇 Continue to Subscribe (繼續以訂閱)。
5. 如果條款和條件可接受，請選擇「接受條款」。訂閱可能需要幾分鐘的時間來處理。
6. 等待感謝訊息出現。此訊息確認您已成功訂閱產品。
7. 在左側導覽窗格中，選擇 [管理訂閱]。此檢視會顯示您所有的訂閱項目。

使用執行個體啟動 AWS 大型主機現代化重新平台 NTT DATA

1. 在 https://console.aws.amazon.com/市集開啟 AWS Marketplace 主控台。
2. 在左側導覽窗格中，選擇 [管理訂閱]。
3. 尋找您AMI要啟動的項目，然後選擇 [啟動新執行個體]。
4. 在「區域」下，選取允許列出的區域。

5. 選擇 [繼續] 以透過啟動EC2。此操作將帶您到 Amazon EC2 控制台。
6. 輸入伺服器的名稱。
7. 選取符合您專案效能和成本需求的執行個體類型。例證大小的建議起點為`c5.2xLarge`。
8. 選擇現有的 key pair，或建立並儲存新的金鑰配對。如需金鑰配對的相關資訊，請參閱 Amazon EC2 使用者指南中的 Amazon EC2 金鑰配對和 Linux 執行個體。
9. 編輯網路設定，並選擇允許列出VPC和適當的子網路。
10. 選擇現有的安全性群組或建立新群組。如果這是企業伺服器 Amazon EC2 執行個體，TCP通常會允許連接埠 86 和 10086 的流量來管理 Micro Focus 組態。
11. 設定 Amazon EC2 執行個體的儲存空間。
12. 檢閱摘要，然後選擇啟動執行個體。若要成功啟動，執行個體類型必須是有效的。如果啟動失敗，請選擇 [編輯執行個體組態] 並選擇不同的執行個體類型。
13. 看到成功訊息後，請選擇 [Connect 至執行個體]。
14. 在打開 Amazon EC2 控制台https://console.aws.amazon.com/ec2/。
15. 在左側導覽窗格的 [執行個體] 功能表下，選擇 [執行個體]。
16. 在主窗格中，檢查執行個體的狀態。

開始使用NTT資料

佈建 Amazon EC2 執行個體之後，請使用使用者名稱SSH進入該執行個體`ec2-user`。屏幕看起來像下面的圖像。

在該`/opt/software/`文件夾下，有一個名為的文件夾`UniKix_Product_Guides`，如下圖所示。

```
[ec2-user@ip-172-          ~]$ ls -l /opt/software/
total 64
lrwxrwxrwx.  1 root     root        23 Oct 17 19:27 BPE -> /opt/software/BPE17.2.3
drwxr-xr-x.  6 ec2-user ec2-user 16384 Oct  4 16:38 BPE17.2.3
lrwxrwxrwx.  1 ec2-user ec2-user    32 Oct 17 19:27 COBOL -> /opt/software/NTT_DATA_COBOL_6.5
drwxr-xr-x. 11 ec2-user ec2-user 16384 Oct 17 19:27 NTT_DATA_COBOL_6.5
lrwxrwxrwx.  1 ec2-user ec2-user    36 Oct 17 19:28 NTT_DATA_TPE_Agent -> /opt/software/NTT_DATA_TPE_Agent_4.9
drwxr-xr-x.  8 ec2-user ec2-user 16384 Nov  9 01:59 NTT_DATA_TPE_Agent_4.9
lrwxrwxrwx.  1 ec2-user ec2-user    25 Oct 17 19:28 Secure -> /opt/software/Secure6.3.1
drwxr-xr-x.  8 ec2-user ec2-user   156 Oct 17 19:28 Secure6.3.1
lrwxrwxrwx.  1 ec2-user ec2-user    23 Oct 17 19:27 TPE -> /opt/software/TPE17.2.2
drwxr-xr-x. 12 ec2-user ec2-user 16384 Oct  4 16:34 TPE17.2.2
lrwxrwxrwx.  1 ec2-user ec2-user    20 Oct 17 19:28 UCM -> /opt/software/UCM2.1
drwxr-xr-x.  7 ec2-user ec2-user   173 Oct 17 19:28 UCM2.1
drwxr-xr-x.  2 ec2-user ec2-user     6 Dec 12 12:20 UniKix_Product_Guides
drwxr-xr-x.  2 ec2-user ec2-user     6 Oct 17 19:22 bin
drwxr-xr-x.  2 ec2-user ec2-user    34 Nov 10 17:03 license
drwxr-xr-x.  8 root     root        88 Oct 17 19:28 staging
```

該`UniKix_Product_Guides`資料夾包含此 Amazon EC2 執行個體上安裝的下列元件的說明文件：

- NTT DATA TPE
- NTT DATA BPE
- NTTDATA企業 COBOL
- NTTDATA UniKix 安全
- NTTDATA UniKix 中央經理

上一個影像中顯示的`software`資料夾具有上述所列元件的二進位檔案。

成功驗證 Amazon EC2 執行個體後，請NTTDATA依照資料文件開始使用 AWS 大型主機現代化重新平台。NTT

瞭解受管理應用程式 AWS Mainframe Modernization

如果您是新手， AWS Mainframe Modernization 請查看以下主題以開始使用：

- 什麼是 AWS 大型主機現代化？
- 設定 AWS 大型主機現代化
- 教程：為 AWS 藍光時代設置託管運行時
- 教學課程：設定 Micro Focus 的受管理執行階段

中的應用程式 AWS Mainframe Modernization 包含已移轉的大型主機工作負載。此應用程式類似於大型主機上的工作負載，並與執行階段環境相關聯。您可以將批次檔案和資料集新增至應用程式，並在應用程式執行時監視它們。您可以為移轉的每個工作負載建立 AWS Mainframe Modernization 應用程式。建立 AWS Mainframe Modernization 應用程式時，您可以指定應用程式在建立應用程式時執行的引擎。如果您使用的是自動重構模式，請選擇 AWS Blu Age，如果您使用的是重構模式，請選擇 Micro Focus。

主題

- 為移轉的應用程式建立 AWS 資源
- 建立 AWS Mainframe Modernization 應用程式
- 部署 AWS Mainframe Modernization 應用程式
- 更新 AWS Mainframe Modernization 應用程式
- 刪除 AWS Mainframe Modernization 應用程式
- 提交應用程式的批次 AWS Mainframe Modernization 工作
- 取消應用程式的批次 AWS Mainframe Modernization 工作
- 匯入 AWS Mainframe Modernization 應用程式的資料集
- 管理 AWS Mainframe Modernization 應用程式的交易
- 設定管理的應用程式
- AWS Mainframe Modernization 應用定義參考
- AWS 大型主機現代化資料集定義參考

為移轉的應用程式建立 AWS 資源

若要在中執行移轉的應用程式 AWS，您必須使用其他 AWS 資源建立一些資源 AWS 服務。您必須建立的資源包括下列項目：

- 用於存放應用程式程式碼、組態、資料檔案和其他必要成品的 S3 儲存貯體。
- 用於保存應用程式所需資料的 Amazon RDS 或亞馬遜 Aurora 資料庫。
- 和 AWS KMS key, 這是需 AWS Secrets Manager 要通過創建和存儲秘密.
- 保存資料庫認證的秘 Secrets Manager 理員。

> **Note**
>
> 每個移轉的應用程式都需要自己的一組這些資源。這是一個最小的設置。您的應用程式可能還需要其他資源，例如 Amazon Cognito 密碼或 MQ 佇列。

所需的許可

請確定您具有下列權限：

- `s3:CreateBucket`, `s3:PutObject`
- `rds:CreateDBInstance`
- `kms:CreateKey`
- `secretsmanager:CreateSecret`

Amazon S3 儲存貯體

重構和重組的應用程式都需要 Amazon S3 儲存貯體，您可以按如下方式設定：

```
bucket-name/root-folder-name/application-name
```

bucket-name

Amazon S3 命名限制範圍內的任何名稱。建議您將 AWS 區域 名稱納入值區名稱中。請務必在計劃部署移轉應用程式的相同區域中建立值區。

root-folder-name

在應用程式定義 (您建立為應用程式的一部分) 中滿足條件約束所需的 AWS Mainframe Modernization 名稱。您可以使`root-folder-name`用區分應用程式的不同版本，例如 V1 和 V2。

application-name

移轉應用程式的名稱，例如， PlanetsDemo 或 BankDemo。

資料庫

重構和重組的應用程式都可能需要資料庫。您必須根據每個執行階段引擎的特定需求建立、設定和管理資料庫。 AWS Mainframe Modernization 支援此資料庫傳輸中的加密。如果您在資料庫SSL上啟用，請確定您在資料庫密碼`sslMode`中指定以及資料庫的連線詳細資訊。如需詳細資訊，請參閱AWS Secrets Manager 秘密。

如果您使用 AWS 藍光時代重構模式，並且需要一個 BluSam 數據庫，則 AWS Blu Age 運行時引擎需要一個 Amazon Aurora Postgre SQL 數據庫，您必須創建，配置和管理該數據庫。該 BluSam 數據庫是可選的。只有在您的應用程式需要時才建立此資料庫。若要建立資料庫，請依照 Amazon Aurora 使用者指南中的建立 Amazon Aurora 資料庫叢集中的步驟執行。

如果您使用的是微焦點重新平台模式，您可以創建一個 Amazon RDS 或 Amazon Aurora Post SQL gre 數據庫。若要建立資料庫，請遵循 Amazon 使用者指南中的建立 Amazon 資料RDS庫執行個體中的步驟，或在 Amazon Aurora RDS 使用者指南中建立 Amazon Aurora 資料庫叢集中的步驟執行。

對於這兩個執行階段引擎，您必須將資料庫認證儲存在 AWS Secrets Manager 使用 AWS KMS key 來加密它們。

AWS Key Management Service 鍵

您必須將應用程式資料庫的認證安全地儲存在中 AWS Secrets Manager。若要在密碼管理員中建立密碼，您必須建立 AWS KMS key. 若要建立金KMS鑰，請依照AWS Key Management Service 開發人員指南中建立金鑰中的步驟進行。

建立金鑰之後，您必須更新金鑰原則以授與 AWS Mainframe Modernization 解密權限。新增下列原則陳述式：

```
{
   "Effect" : "Allow",
   "Principal" : {
   "Service" : "m2.amazonaws.com"
```

```
   },
   "Action" : "kms:Decrypt",
   "Resource" : "*"
   }
```

AWS Secrets Manager 秘密

您必須將應用程式資料庫的認證安全地儲存在中 AWS Secrets Manager。若要建立密碼，請遵循《AWS Secrets Manager 使用者指南》中的建立資料庫密碼中的步驟進行操作。

AWS Mainframe Modernization 支援此資料庫傳輸中的加密。如果您在資料庫SSL上啟用，請確定您在資料庫密碼`sslMode`中指定以及資料庫的連線詳細資訊。您可以為下列其中一個值指定`sslMode`：`verify-full`、`verify-ca`、或`disable`。

在金鑰建立程序期間，選擇 [資源權限-選用]，然後選擇 [編輯權限]。在策略編輯器中，新增以資源為基礎的策略，如下所示，以擷取加密欄位的內容。

```
{
   "Effect" : "Allow",
   "Principal" : {
   "Service" : "m2.amazonaws.com"
   },
   "Action" : "secretsmanager:GetSecretValue",
   "Resource" : "*"
   }
```

建立 AWS Mainframe Modernization 應用程式

使用主 AWS Mainframe Modernization 控台建立 AWS Mainframe Modernization 應用程式。建立應用程式可讓您使用已移轉的大型主機工作負載來執行工作。

這些說明假設您已完成設定 AWS 大型主機現代化中的步驟。

建立應用程式

建立應用程式

1. 在開啟 AWS Mainframe Modernization 主控台https://console.aws.amazon.com/m2/。
2. 在選取 AWS 區域 器中，選擇您要建立應用程式的「區域」。
3. 在 Applications (應用程式) 頁面上，選擇 Create application (建立應用程式)。

4. 在 [指定基本資訊] 頁面的 [名稱和說明] 區段中，輸入應用程式的名稱。
5. (選擇性) 在應用程式說明欄位中，輸入應用程式的說明。此說明可協助您和其他使用者識別應用程式的用途。
6. 在「引擎類型」區段中，選擇「藍光時代」進行自動重構，或選擇 Micro Focus 進行重構。
7. 如果您要使用客戶管理的KMS金鑰，請在「金 AWS KMS 鑰」區段中選擇「自訂加密設定」。如需詳細資訊，請參閱AWS 大型主機現代化服務的靜態資料加密。

   > **Note**
   >
   > 根據預設， AWS Mainframe Modernization 會使用為您 AWS Mainframe Modernization 擁有和管理的 AWS KMS 金鑰來加密資料。但是，您可以選擇使用客戶管理的 AWS KMS 金鑰。

8. (選擇性) 依名稱或 Amazon 資源名稱選擇金 AWS KMS 鑰 (ARN)，或選擇建立 AWS KMS 金鑰移至 AWS KMS 主控台並建立新 AWS KMS 金鑰。
9. (選擇性) 在「標記」區段中，選擇「新增標記」，將一或多個應用程式標記新增至您的應用程式。應用程序標籤是一個自定義屬性標籤，可幫助您組織和管理 AWS 資源）。
10. 選擇 Next (下一步)。
11. 在 [資源和設定] 區段中，使用內嵌編輯器輸入應用程式定義。或者，選擇在 Amazon S3 儲存貯體中使用應用程式定義JSON檔案，並提供您要使用的應用程式定義位置。如需詳細資訊，請參閱 AWS 藍光時代的應用程序定義 或 微焦點應用定義。
12. 選擇 Next (下一步)。
13. 在 [複查並建立] 頁面上，複查您輸入的資訊，然後選擇 [建立應用程式]。

部署 AWS Mainframe Modernization 應用程式

使用主 AWS Mainframe Modernization 控台部署 AWS Mainframe Modernization 應用程式。執行工作之前，您必須先在執行階段環境中部署應用程式。

這些說明假設您已完成設定 AWS 大型主機現代化中的步驟。

部署應用程式

若要執行 AWS Mainframe Modernization 應用程式，您必須先將其部署到執行階段環境。一個應用程式可以有多個版本。每個應用程式版本都有自己的應用程式定義。若要部署應用程式，您必須指定要部署的版本。

您一次只能部署指定應用程式的一個版本。如果您部署應用程式的版本，然後決定改為部署不同的版本，則必須先停止應用程式 (如果應用程式正在執行)。

若要部署應用程式

1. 在開啟 AWS Mainframe Modernization 主控台https://console.aws.amazon.com/m2/。
2. 在選取 AWS 區域 器中，選擇您要建立應用程式的「區域」。
3. 在 [應用程式] 頁面上，選擇您要部署的應用程式。
4. 選擇部署應用程式。
5. 在 [可用的版本] 區段中，選擇您要部署的版本。
6. 在「環境」段落中，選擇您要執行應用程式的程式實際執行環境。
7. 選擇部署。

若要部署不同版本的已部署應用程式

1. 在開啟 AWS Mainframe Modernization 主控台https://console.aws.amazon.com/m2/。
2. 在選取 AWS 區域 器中，選擇您要建立應用程式的「區域」。
3. 在 [應用程式] 頁面上，選擇您要部署的應用程式。
4. 從「動作」功能表選擇「停止應用程式」。
5. 應用程式停止後，選擇部署應用程式。
6. 在 [可用的版本] 區段中，選擇您要部署的版本。在「環境」段落中，會預先選取應用程式已經建置在其中的環境。
7. 選擇部署。

更新 AWS Mainframe Modernization 應用程式

使用主 AWS Mainframe Modernization 控台更新 AWS Mainframe Modernization 應用程式。更新應用程式會建立應用程式的新版本。

這些說明假設您已完成設定 AWS 大型主機現代化中的步驟。

更新應用程式

AWS Mainframe Modernization 應用程式可以有多個版本，每個版本都有自己的應用程式定義。若要更新應用程式，請提供新的應用程式定義。這將創建應用程序的新版本。

更新應用程式

1. 在開啟 AWS Mainframe Modernization 主控台https://console.aws.amazon.com/m2/。
2. 在選取 AWS 區域 器中，選擇建立您要更新之應用程式的「區域」。
3. 在 [應用程式] 頁面上，選擇您要更新的應用程式。
4. 在應用程式詳細資訊頁面的 [目前定義] 區段中，選擇 [編輯] 以更新目前的應用程式定義。
5. 在 [更新應用程式] 頁面上，使用內嵌編輯器更新目前的應用程式定義。

   或者，選擇在 Amazon S3 儲存貯體中使用應用程式定義JSON檔案，並提供您要使用的應用程式定義位置。如需詳細資訊，請參閱 AWS 藍光時代的應用程序定義 或 微焦點應用定義。
6. 當您完成應用程式定義的更新時，請選擇 [更新]。

> **Note**
>
> 更新應用程式之後，您必須再次部署它。如需詳細資訊，請參閱部署 AWS Mainframe Modernization 應用程式。

刪除 AWS Mainframe Modernization 應用程式

您可以使用主控台從環境中刪除 AWS Mainframe Modernization 應用程 AWS Mainframe Modernization 式。

這些說明假設您已完成設定 AWS 大型主機現代化中的步驟。

刪除應用程式

如果您需要刪除某個 AWS Mainframe Modernization 應用程式，且該應用程式正在執行，請務必先停止該應用程式。您可以在 [應用程式] 頁面上查看應用程式狀態。

如欲刪除應用程式

1. 在開啟 AWS Mainframe Modernization 主控台https://console.aws.amazon.com/m2/。
2. 在選取 AWS 區域 器中，選擇您要從環境中刪除的應用程式建立的區域。
3. 在 [應用程式] 頁面上，選擇要從環境中刪除的應用程式，然後選擇 [動作]。
4. (選擇性) 如果應用程式的狀態為`Running`，請選擇停止應用程式。
5. 選擇「從環境中刪除」。

刪除程序會立即開始。

提交應用程式的批次 AWS Mainframe Modernization 工作

在中， AWS Mainframe Modernization 您可以提交應用程式的批次工作。您可以提交或取消批次工作，以及檢閱有關批次工作執行的詳細資訊。每次提交批次工作時，都 AWS Mainframe Modernization 會建立個別的批次工作執行。您可以監視此工作執行。您可以依名稱和批次工作的提供JCL或指令碼檔案來搜尋批次工作。

> **Important**
>
> 如果您取消批次處理工作，則不會刪除該工作。它取消了批處理作業的特定運行。批次工作記錄仍可供您在批次工作執行的詳細資訊中檢視。

如果批次工作需要存取一或多個資料集，請使用 AWS Mainframe Modernization 控制台匯入資料集。如需詳細資訊，請參閱匯入 AWS Mainframe Modernization 應用程式的資料集。

這些指示假設您已完成中設定 AWS 大型主機現代化和中的步驟建立 AWS Mainframe Modernization 應用程式。

主題

- 提交批次工作
- 重新啟動批次工作

提交批次工作

若要提交批次工作

1. 在開啟 AWS Mainframe Modernization 主控台https://console.aws.amazon.com/m2/。
2. 在選取 AWS 區域 器中，選擇您要提交批次工作之應用程式的建立區域。
3. 在 [應用程式] 頁面上，選擇您要送出批次工作的應用程式。

> **Note**
>
> 您必須先成功部署應用程式，才能將批次工作提交至應用程式。

4. 在應用程式詳細資訊頁面上，選擇 Batch 工作。
5. 選擇 Submit job (提交任務)。
6. 在「選取指令碼」區段中，選擇指令碼。您可以依名稱搜尋所需的指令碼。
7. 選擇 Submit job (提交任務)。

重新啟動批次工作

重新啟動批次工作

> **Important**
>
> 批次工作重新啟動僅適用於 MicroFocus 環境引擎 8.0.6 或更高版本。您也需要將EFS或FSx檔案系統附加至您的環境。

1. 在開啟 AWS Mainframe Modernization 主控台https://console.aws.amazon.com/m2/。
2. 在選取 AWS 區域 器中，選擇建立應用程式和批次工作的區域。
3. 在 [應用程式] 頁面上，選擇要重新啟動批次工作的應用程式。
4. 在應用程式詳細資訊頁面上，選擇 Batch 工作。
5. 從產生的清單中選取要重新啟動的批次工作。瀏覽至「動作」功能表，然後選擇「重新啟動工作」
6. 指定重新啟動批次工作的方式。您可以選擇從頭開始重新啟動，或使用步驟或程序重新啟動。
   - 「從頭開始重新啟動」選項可讓您從頭開始重新啟動批次工作的所有步驟。
   - 使用「使用步驟或處理步驟重新啟動」選項，您可以選擇要重新啟動的特定步驟或程序 (程序步驟)，並選擇性地選擇要在其後結束的步驟或程序。

   > **Note**
   >
   > 結束步驟或程序步驟必須大於或等於開始步驟或處理步驟編號。

7. 選擇 Submit job (提交任務)。

取消應用程式的批次 AWS Mainframe Modernization 工作

在中， AWS Mainframe Modernization 您可以取消應用程式的批次工作。您可以檢閱有關批次工作執行的詳細資訊。每次提交批次工作時，都 AWS Mainframe Modernization 會建立個別的批次工作執行。您可以監視此工作執行。您可以依名稱和批次工作的提供JCL或指令碼檔案來搜尋批次工作。

> **Important**
>
> 如果您取消批次處理工作，則不會刪除該工作。它取消了批處理作業的特定運行。批次工作記錄仍可供您在批次工作執行的詳細資訊中檢視。

取消批次處理工作

當您取消批次工作時，它不會刪除批次工作，而是刪除該批次工作的工作執行。您仍然可以檢視批次處理工作的詳細資料。

若要取消批次工作

1. 在開啟 AWS Mainframe Modernization 主控台https://console.aws.amazon.com/m2/。
2. 在選取 AWS 區域 器中，針對批次工作選擇應用程式的「區域」。
3. 從批次工作清單中尋找並選取您要取消的批次工作。
4. 選擇「作業」，並選擇「取消工單」。
5. 選擇「取消批次工作」。

這將取消您已排定執行的任何批次工作任務。

匯入 AWS Mainframe Modernization 應用程式的資料集

AWS Mainframe Modernization 您可以使用匯入要與應用程式搭配使用的資料集。您可以在 Amazon S3 儲存貯體中存放的JSON檔案中指定資料集，也可以分別指定資料集組態值。匯入資料集之後，您可以檢閱匯入任務的詳細資訊，以確認您想要的資料集已匯入。應用程式的所有目錄資料集會一起列在主控台中。

使用主 AWS Mainframe Modernization 控台匯入應用 AWS Mainframe Modernization 程式的資料集。

這些指示假設您已完成中設定 AWS 大型主機現代化和中的步驟建立 AWS Mainframe Modernization 應用程式。

匯入資料集

匯入資料集的步驟

1. 在開啟 AWS Mainframe Modernization 主控台https://console.aws.amazon.com/m2/。
2. 在選取 AWS 區域 器中，選擇建立您要匯入資料集之應用程式的「區域」。
3. 在 [應用程式] 頁面上，選擇您要匯入資料集的應用程式。
4. 在應用程式詳細資訊頁面上，選擇 [資料集]。
5. 選擇匯入。
6. 執行以下任意一項：
   - 選擇在 Amazon S3 儲存貯體中使用資料集組態JSON檔案，並提供資料集組態的位置。
   - 選擇使用引導式組態分別指定資料集組態值。如需特the section called “資料集定義參考”定定義的詳細資訊，請

     輸入名稱、資料集組織 (VSAM、PO GDG、PS)、位置和外部 Amazon S3 位置，以及每個資料集組態值的參數設定。在引導式組態中，您也可以選擇「產生」JSON 來檢閱輸入的JSON組態。
7. 選擇提交。

管理 AWS Mainframe Modernization 應用程式的交易

AWS Mainframe Modernization 您可以透過要求執行應用程式，與許多其他使用者提交要求以使用相同檔案和程式執行相同應用程式的同時執行應用程式。單一交易包含執行所需處理的一或多個應用程式。

這些指示假設您已完成中設定 AWS 大型主機現代化和中的步驟建立 AWS Mainframe Modernization 應用程式。

管理應用程式的交易

若要管理應用程式的交易

1. 在開啟 AWS Mainframe Modernization 主控台https://console.aws.amazon.com/m2/。
2. 在選取 AWS 區域 器中，選擇建立您要執行之應用程式的區域。
3. 在「應用程式」頁面上，選擇您要管理交易的應用程式。

4. 在「交易」頁標的「異動資源」下，從下拉式清單中選擇您希望資源的顯示方式。您可以根據交易資源、群組、清單或來顯示資源SITs。

   - 交易資源可讓您根據檔案定義、交易定義、程式定義或暫時性資料佇列定義來選擇資源類型。

     > **Note**
     >
     > 該 AWS Mainframe Modernization 服務支持其他資源類型來管理應用程序的事務，並且可以在控制台中訪問。

   - 組是事務資源的集合。您可以選擇要與異動資源產生關聯的群組。
   - 列表是組的有序集合。您可以在清單檢視中查看所有交易資源和群組。啟動清單會決定伺服器初始化時要載入的資源。
     - 使用 AWS Blu Age 重構引擎，您可以指定要在啟動時包含的列表。列表數量沒有限制。
     - 使用 Micro Focus 重新平台引擎，您最多可以在一個清單中指定四個SIT清單。
   - SIT(系統初始化表格) 顯示所有可用的交易組態。您可以SITs根據屬性（名稱，描述和啟動列表）查找。您也可以選擇與您選擇的列表相關聯SIT。

     > **Note**
     >
     > SITs僅適用於微焦點重新平台引擎。

5. 選擇異動資源以顯示所有資源資訊。您也可以檢視與異動資源相關的所有屬性。

設定管理的應用程式

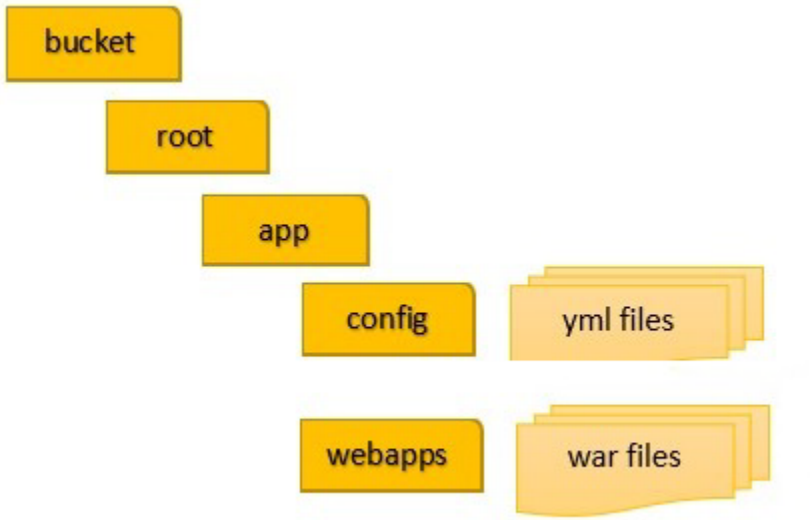
您可以將應用程式設定為包含舊式公用程式的存取權。您也可以自訂其他屬性。若要瞭解您可以設定的項目和位置，請參閱the section called “AWS 藍光時代管理應用程式的結構”本節以瞭解 AWS Blu Ege 現代化應用程式的整體結構。

主題

- AWS 藍光時代管理應用程式的結構
- 設定受管理應用程式之公用程式的
- 使用 AWS Blu Age 引擎為受管理的應用程序添加配置屬性

AWS 藍光時代管理應用程式的結構

如果您使用 AWS Blu Age 重構模式，則 AWS Blu Age 運行時引擎會在 S3 存儲桶中的`application-name`文件夾內預期以下結構：



config

包含專YAML案的檔案。這些是應用程式特定的YAML檔案，通常命名為類似大型主機現代化提供`application-planetsdemo.yaml`和設定的`application-main.yaml`檔案，而不是 AWS 大型主機現代化為您自動提供和設定的檔案。

Web 應用程式

包含您應用程式的`war`檔案。這些文件是現代化過程的輸出。

一個應用程序也可以有以下可選文件夾：

小/SQL

包含為您的應用程式初始化資JICS料庫的`initJics.sql`指令碼。

指令碼

包含應用程式指令碼，您也可以直接在`war`檔案內提供。

sql

包含應用程式SQL檔案，您也可以直接在檔案內提供這些`war`檔案。

LNK

包含應用程式LNK檔案，您也可以直接在檔案內提供這些war檔案。

額外的

包含可為現代化應用程式提供其他功能的 jar。

管理應用程式的 Java 選項

若要管理應用程式的某些 Java 選項，請將名為的屬性檔案新增tomcat.properties至資application-name料夾。此檔案可以有三個屬性：xms指定 Java 記憶體耗用量下限xmx、指定 Java 記憶體耗用量上限，以及dnscachettl管理 dns 解析度的快取持續時間。以下是有效tomcat.properties檔案內容的範例。

```
xms=512M
xmx=1G
dnscachettl=5
```

您為前兩個性質指定的值可以採用下列任一單位：

- 位元組：不指定單位。
- 千位元組：將 K 附加至值。
- 百萬位元組：將 M 附加至值。
- 千兆字節：將 G 附加到值。

第三個屬性的值代表快取持續時間 (以秒為單位)，值可以是 -1 (永遠快取)，或者範圍從 0 (永遠不會快取) 到 999。在受管理的應用程式部署環境中，預設值為 -1。

設定受管理應用程式之公用程式的

當您使用 AWS Blu Age 重構大型主機應用程式時，如果您的應用程式依賴於各種舊版平台公用程式，您可能需要為IDCAMS各種舊版平台公用程式提供支援，例如、、等。INFUTILB SORT AWS Blu Age 重構通過與現代化應用程序一起部署的專用 Web 應用程序提供了此訪問權限。此 Web 應用程式需要您必須提供的組態檔案。application-utility-pgm.yml如果您未提供此設定檔，則 Web 應用程式無法與您的應用程式一起部署，因此無法使用。

主題

- 組態屬性

本主題說明您可以在application-utility-pgm.yml組態檔案中指定的所有可能屬性及其預設值。本主題說明必要和選用的屬性。下面的例子是一個完整的配置文件。它會依照我們建議的順序列出屬性。您可以使用此範例作為您自己組態檔案的起點。

```
# If the datasource support mode is not static-xa, spring JTA transactions
 autoconfiguration must be disabled
 spring.jta.enabled: false
 logging.config: 'classpath:logback-utility.xml'

 # Encoding
 encoding: cp1047

 # Encoding to be used by INFUTILB and DSNUTILB to generate and read SYSPUNCH files
 sysPunchEncoding: cp1047

 # Utility database access
 spring.aws.client.datasources.primary.secret: `arn:aws:secretsmanager:us-
west-2:111122223333:secret:business-FfmXLG`

 treatLargeNumberAsInteger: false

 # Zoned mode : valid values = EBCDIC_STRICT, EBCDIC_MODIFIED, AS400
 zonedMode: EBCDIC_STRICT

 jcl.type: mvs

 # Unload properties
 # For date/time: if use database configuration is enabled, formats are ignored
 # For nbi; use hexadecimal syntaxe to specify the byte value
 unload:
  sqlCodePointShift: 384
  nbi:
    whenNull: "6F"
    whenNotNull: "00"
  useDatabaseConfiguration: false
  format:
    date: MM/dd/yyyy
    time: HH.mm.ss
    timestamp: yyyy-MM-dd-HH.mm.ss.SSSSSS
  chunkSize:500
  fetchSize: 500
  varCharIsNull: false
  columnFiller: space
```

```
# Load properties
# Batch size for DSNUTILB Load Task
load:
 sqlCodePointShift: 384
 batchSize: 500
 format:
   localDate: dd.MM.yyyy|dd/MM/yyyy|yyyy-MM-dd
   dbDate: yyyy-MM-dd
   localTime: 'HH:mm:ss|HH.mm.ss'
   dbTime: 'HH:mm:ss'

table-mappings:
 TABLE_1_NAME : LEGACY_TABLE_1_NAME
 TABLE_2_NAME : LEGACY_TABLE_2_NAME
```

組態屬性

您可以在組態檔案中指定下列屬性。

春天. 啟用

(選擇性) 控制是否啟用JTA支援。對於公用程式，建議您將此值設定為false。

```
spring.jta.enabled : false
```

記錄. 配置

(必要) 指定專用記錄器組態檔的路徑。我們建議您使用名稱logback-utility.xml並提供此檔案做為現代化應用程式的一部分。組織這些文件的常用方法是將所有記錄器配置文件放在同一個位置，通常位於包含YAML配置文件的文件夾中的子/config/logback文件夾中。/config如需詳細資訊，請參閱 Logback 文件中的第 3 章：登入設定。

```
logging.config : classpath:logback-utility.xml
```

編碼

(必要) 指定公用程式使用的字元集。在大多數情況下，當您從 z/OS 平台移轉時，此字元集是一EBCDIC種變體，而且應該符合針對現代化應用程式設定的字元集。如果未設定，則預設值為ASCII。

```
encoding : cp1047
```

sysPunchEncoding

(選擇性) 指定DSNUTILB用來產生INFUTILB和讀取SYSPUNCH檔案的字元集。如果您依原樣使用舊版平台中的SYSPUNCH檔案，則此值應為EBCDIC變體。如果未設定，則預設值為ASCII。

```
sysPunchEncoding : cp1047
```

資料來源組態

某些資料庫相關公用程式 (例如LOAD和UNLOAD) 需要透過資料來源存取目標資料庫。與 AWS 大型主機現代化中的其他資料來源定義一樣，此存取需要您使用。 AWS Secrets Manager指向「密碼管理員」中適當密碼的屬性如下：

主要資料來源

這是主要的商務應用程式資料庫。

客戶端. 數據搜索. 主要. 秘密

(選擇性) 指定秘 Secrets Manager 中包含資料來源屬性的密碼。

```
spring.aws.client.datasources.primary.secret: datasource-secret-ARN
```

用戶端. 資料搜尋. 主要的 .db 名稱

(選擇性) 指定目標資料庫名稱，如果資料庫名稱不是直接在資料庫密碼中提供的，並具有dbname屬性。

```
spring.aws.client.datasources.primary.dbname: target-database-name
```

春天. aws.w. 客戶端. 數據搜索. 主要類型

(選擇性) 指定要使用之連線集區實作的完整名稱。預設值為com.zaxxer.hikari.HikariDataSource。

```
spring.aws.client.datasources.primary.type: target-datasource-type
```

如果主資料來源的類型為com.zaxxer.hikari.HikariDataSource，您可以指定其他屬性，如下所示：

春天. 數據搜索. 主要. [屬性名稱]

(選擇性) 您可以使用此格式來指定配置主要資料來源連線集區實作的額外屬性。

以下是主資料來源類型的範例com.zaxxer.hikari.HikariDataSource。

```
spring:
   datasource:
       primary:
         autoCommit: XXXX
         maximumPoolSize: XXXX
         keepaliveTime: XXXX
         minimumIdle: XXXX
         idleTimeout: XXXX
         connectionTimeout: XXXX
         maxLifetime: XXXX
```

其他公用事業數據源

除了主資料來源之外，您還可以提供其他公用程式資料來源。

春天. aws.w 客戶端. 實用程序.

(選擇性) 指定公用程式資料來源名稱清單。

```
spring.aws.client.utility.pgm.datasources.names: dsname1, dsname2, dsname3
```

應用程式 .pgm. 資料來源. [dsname]. 秘密

(選擇性) 指定主控資料來源屬性的密ARN碼。SSM在中指定的名稱清單中提供 [dsname]。spring.aws.client.utility.pgm.datasources.names

```
spring.aws.client.utility.pgm.datasources.dsname1.secret: datasource-secret-ARN
```

應用程式 .pgm. 資料來源. [dsname]. 資料庫名稱

(選擇性) 如果未使用屬性直接在資料庫密碼中提供資料庫名稱，則指定目標資料庫名dbname稱。在中指定的名稱清單中提供 [dsname]。spring.aws.client.utility.pgm.datasources.names

```
spring.aws.client.utility.pgm.datasources.dsname1.dbname: target-database-name
```

應用程式 .pgm. 資料來源. [dsname]. 類型

(選擇性) 指定要使用之連線集區實作的完整名稱。預設值為 com.zaxxer.hikari.HikariDataSource。在中指定的名稱清單中提供 [dsname]。spring.aws.client.utility.pgm.datasources.names

```
spring.aws.client.utility.pgm.datasources.dsname1.type: target-datasource-type
```

如果公用程式資料來源類型為com.zaxxer.hikari.HikariDataSource，您可以提供其他屬性，如下所示：

春天. 數據源。 [dsname]。 [屬性名稱]

(選擇性) 指定額外屬性的集合，以配置公用程式資料來源連線集區實作。在中指定的名稱清單中提供 [dsname]。spring.aws.client.utility.pgm.datasources.names以下列格式指定屬性：property_name : value

以下是其他公用程式資料來源類型的範例com.zaxxer.hikari.HikariDataSource：

```
spring:
   datasource:
       dsname1:
         connectionTimeout: XXXX
         maxLifetime: XXXX
       dsname2:
         connectionTimeout: XXXX
         maxLifetime: XXXX
       dsname3:
         connectionTimeout: XXXX
         maxLifetime: XXXX
```

treatLargeNumberAsInteger

(選擇性) 與 Oracle 資料庫引擎特定與DSNTEP2/DSNTEP4公用程式使用相關。如果將此旗標設定為 true，則來自 Oracle 資料庫 (NUMBER(38,0)) 的大數字會被視為整數。預設：false

```
treatLargeNumberAsInteger : false
```

zonedMode

(選擇性) 將分區模式設定為編碼或解碼分區資料類型。此設定會影響符號數字的表示方式。有效值如下：

- EBCDIC_ STRICT：預設值。使用嚴格的標誌處理定義。根據字元集是否為EBCDIC或ASCII，符號數字表示會使用下列字元：
  - EBCDIC對應於 bytes (Cn+Dn) 的字元，以表示正數和負數範圍 (+0to+9, -0 to-9)。字符顯示為{，A到 I}，J到 R
  - ASCII對應於 bytes (3n+7n) 的字元，以表示正數和負數範圍 (+0to+9, -0 to-9)。字符顯示0為9，p到 y
- EBCDIC_ MODIFIED：使用修改過的定義進行符號處理。對於EBDIC和ASCII，相同的字符列表代表符號數字，即+9映射+0A到 { + I 和-9映射-0到 } + J 到R。 \
- AS400：用於來自 iSeries (AS400) 個平台的現代化舊資產。

```
zonedMode:EBCDIC_STRICT
```

jcl.type

(選擇性) 指出現代化指JCL令碼的舊版類型。如果叫IDCAMS用類vse型為，公用程式會使用此設定來調JCL整傳回碼。有效值如下：

- mvs (預設值)
- vse

```
jcl.type : mvs
```

資料庫卸載公用程式相關屬性

使用這些性質來規劃將資料庫表格卸載至資料集的公用程式。以下所有屬性都是可選的。

此範例顯示所有可能的卸載屬性。

```
# Unload properties
 # For date/time: if use database configuration is enabled, formats are ignored
 # For nbi; use hexadecimal syntaxe to specify the byte value
 unload:
 sqlCodePointShift: 0
 nbi:
 whenNull: "6F"
 whenNotNull: "00"
 useDatabaseConfiguration: false
 format:
 date: MM/dd/yyyy
```

```
time: HH.mm.ss
timestamp: yyyy-MM-dd-HH.mm.ss.SSSSSS
chunkSize: 0
fetchSize: 0
varCharIsNull: false
columnFiller: space
```

sqlCodePoint移位

(選擇性) 指定整數值，代表資料上使用的SQL代碼點偏移。預設值為 0。這意味著沒有代碼點移位進行。將此設定與用於現代化應用程式的程式SQL碼點偏移參數對齊。使用程式碼點移位時，此參數最常用的值為 384。

```
unload.sqlCodePointShift: 0
```

nbi

(選擇性) 指定空指示器位元組。這是新增至資料值右側的十六進位值 (以字串形式)。兩個可能的值如下所示：

- whenNull：當數據值為 null 時添加十六進制值。預設值為 6`。有時會使用高FF值來代替。

  ```
  unload.nbi.whenNull: "6F"
  ```

- whenNotNull：當資料值不為 null 時，新增十六進位值，但資料行可為空。預設值為 00 (低值)。

  ```
  unload.nbi.whenNotNull: "00"
  ```

useDatabaseConfiguration

(選擇性) 指定日期和時間格式屬性。這是用來處理UNLOAD查詢中的日期/時間對象。預設值為 `false`。

- 如果設定為 `true` pgmDateFormatpgmTimeFormat，則使用主組態檔案 (`application-main.yml`) 中的、和pgmTimestampFormat屬性。
- 如果設定為`false`，會使用下列日期和時間格式屬性：
  - `unload.format.date`：指定日期格式化樣式。預設值為 `MM/dd/yyyy`。
  - `unload.format.time`：指定時間格式化樣式。預設值為 `HH.mm.ss`。
  - `unload.format.timestamp`: 指定時間戳記格式化樣式。預設值為 `yyyy-MM-dd-HH.mm.ss.SSSSSS`。

chunkSize

(選擇性) 指定用於建立資料集的SYSREC資料區塊大小。這些數據集是數據集卸載操作的目標，具有 parallel 操作。預設值為 0 (無區塊)。

```
unload.chunkSize:0
```

fetchSize

(選擇性) 指定資料擷取大小。該值是使用資料區塊策略時，一次要擷取的記錄數目。預設：0。

```
unload.fetchSize:0
```

varCharIs空值

(選擇性) 指定如何處理含有空白內容的不可為空的 varchar 資料行。預設值為 false。

如果將此值設定為true，則會將欄內容視為空字串以供卸載使用，而非單一空格字串。僅true針對 Oracle 資料庫引擎案例，將此旗標設為。

```
unload.varCharIsNull: false
```

columnFiller

(選擇性) 指定用來填補 varchar 資料行中已卸載資料欄的值。可能的值為空格或低值。預設值為空格。

```
unload.columnFiller: space
```

資料庫載入相關屬性

使用這些特性來配置將資料集記錄載入目標資料庫的公用程式，例如，DSNUTILB。以下所有屬性都是可選的。

此範例顯示所有可能的負載屬性。

```
# Load properties
 # Batch size for DSNUTILB Load Task
 load:
 sqlCodePointShift: 384
```

```
batchSize: 500
format:
localDate: dd.MM.yyyy|dd/MM/yyyy|yyyy-MM-dd
dbDate: yyyy-MM-dd
localTime: HH:mm:ss|HH.mm.ss
dbTime: HH:mm:ss

table-mappings:
TABLE_1_NAME : LEGACY_TABLE_1_NAME
TABLE_2_NAME : LEGACY_TABLE_2_NAME
```

sqlCodePoint移位

(選擇性) 指定整數值，SQL代表用於資料的代碼點偏移。默認為 0，這意味著應用程序不進行代碼點移動。將此設定與用於現代化應用程式的程式SQL碼點偏移參數對齊。使用代碼點偏移時，此參數最常用的值為 384。

```
load.sqlCodePointShift : 384
```

batchSize

(選擇性) 指定整數值，代表傳送實際批次陳述式至資料庫之前要處理的記錄數目。默認為 0。

```
load.batchSize: 500
```

格式

(選擇性) 指定資料庫載入作業期間用於日期/時間轉換的日期和時間格式化模式。

- `load.format.localDate`：本地日期格式化模式。預設值為 `dd.MM.yyyy|dd/MM/yyyy|yyyy-MM-dd`。
- `load.format.dbDate`: 資料庫日期格式化模式。預設值為 `yyyy-MM-dd`。
- `load.format.localTime`：本地時間格式化模式。預設值為 `HH:mm:ss|HH.mm.ss`。
- `load.format.dbTime`：資料庫時間格式化模式。預設值為 `HH:mm:ss`。

表格對映

(選擇性) 指定舊版與現代表格名稱之間客戶提供的對應集合。DSNUTILB公用程式會消耗這些對映。

以下列格式指定值：MODERN_ TABLE _NAME: LEGACY _ TABLE _ NAME

請見此處範例：

```
table-mappings:
 TABLE_1_NAME : LEGACY_TABLE_1_NAME
 TABLE_2_NAME : LEGACY_TABLE_2_NAME
 ...
 TABLE_*N*_NAME : LEGACY_TABLE_*N*_NAME
```

> **Note**
>
> 當公用程式應用程式啟動時，它會明確記錄所有提供的對應。

使用 AWS Blu Age 引擎為受管理的應用程序添加配置屬性

您可以在重構應用程序的文件`config`夾中添加文件，以使您可以訪問 AWS Blu Age 運行時引擎中的新功能。您必須命名此檔案`user-properties.yml`。該文件不會替換應用程序定義，但擴展它。本主題說明您可以包含在`user-properties.yml`檔案中的屬性。

> **Note**
>
> 您無法變更某些參數，因為它們是由 AWS 大型主機現代化或應用程式定義所控制。在應用程式定義中為您的應用程式定義的所有參數的優先順序高於您在中指定的參數`user-properties.yml`。

如需有關重構應用程式結構的詳細資訊，請參閱。AWS 藍光時代管理應用程式的結構

下圖顯示 AWS Blu Age 範例應用程式結構內`user-properties.yml`檔案的位置 PlanetsDemo。

```
PlanetsDemo-v1/
   ## config/
   #  ## application-PlanetsDemo.yml
   #  ## user-properties.yml
   ## jics/
   ## webapps/
```

組態屬性參考

這是可用屬性的清單。所有參數都是選用的。

主題

- 間隙行走應用程式性
- 間隙步道批次腳本性質
- 加普沃克布魯根住宿
- 間隙行走 CL 指令性質
- 間隙走道 CL 料道性質
- 間隙步道屬JHDB性
- 間隙步道屬JICS性
- 間隙行走運行時屬性
- 間隙行走公用程序屬性
- 其他屬性

間隙行走應用程式性

藍光。 fileLoading。 commitInterval

選用。藍色的SAM提交間隔。

類型：數字

預設值：

卡. 編碼

選用。卡編碼：與`useControlMVariable`.

類型：字串

預設值：CP1145

檢查檔案大小

選用。指定檔案大小是否為記錄大小的倍數，是否要釋放檢查。

類型：布林值

預設：false

數據庫. 游標. 溢出. 允許

選用。指定是否允許游標溢位。設定true為在游標上執行下一個呼叫，無論其位置為何。設定false為可在對游標執行下一次呼叫之前檢查游標是否在最後一個位置。只有在游標為SCROLLABLE (SENSITIVE或INSENSITIVE) 時才啟用

類型：布林值

預設：true

dataSimplifier。 onInvalidNumeric資料

選用。解碼無效數字數據時如何做出反應。允許的值為rejecttoleratespaces、、toleratespaceslowvalues、toleratemost。

類型：字串

預設值：拒絕

defaultKeepExisting。文件

選用。指定是否設定資料集預設先前值。

類型：布林值

預設：false

處置. 檢查存在

選用。指定是否要針對使用DISPSHR或的資料集核發檔案是否存在的檢查OLD。

類型：布林值

預設：false

externalSort. 閾值

選用。排序閾值：何時切換到外部（合併）排序。

類型：字串

預設：null

```
externalSort.threshold: 12MB
```

力量

選用。指定是否在主控台或檔案輸出上使用「人類可讀SYSPRINT」。

類型：布林值

預設：false

forcedDate

選用。強制數據庫中的特定日期和時間。僅在開發和測試期間使用。

預設：null

```
forcedDate: 2022-08-26T12:59:58.123456+01:57
```

frozenDate

選用。凍結資料庫中的日期和時間。僅在開發和測試期間使用。

預設：false

```
frozenDate: false
```

即時消息。 extendedSize

選用。指定是否設定 extendedSize on IMS 訊息。

類型：布林值

預設：false

lockTimeout

選用。無法在指定的時間範圍內取得鎖定時，交易的逾時 (以毫秒為單位)。

類型：數字

預設：500

mapTransfo. 前綴

選用。轉換 ControlM 變量時要使用的前綴列表。每一個用逗號分隔。

類型：字串

預設值：&，@，%

查詢。 useConcatCondition

選用。指定鍵條件是否由密鑰串聯或不構建。

類型：布林值

預設：false

rollbackOnRTE

選用。指定是否針對執行階段例外狀況復原隱含執行單元交易。

類型：布林值

預設：false

sctThreadLimit

選用。觸發指令碼的執行緒限制。

類型：數字

預設：5

sqlCodePoint移位

選用。SQL 代碼點移位。將傳統 rdbms 數據遷移到現代 rdbms 時可能遇到的控制字符的代碼點移動。例如，您可以指定384要符合 unicode 字元\u0180。

類型：數字

預設：0

sqlIntegerOverflow允許

選用。指定是否允許SQL整數溢位，表示是否允許在 host 變數中放置較大的值。

類型：布林值

預設：false

stepFailWhen異常結束

選用。指定是否在步驟失敗或完成執行時引發異常結束。

類型：布林值

預設：true

stopExecutionWhenProgNotFound

選用。指定如果找不到程式，是否停止執行。如果設定為`true`，則在找不到程式時中斷執行。

類型：布林值

預設：true

uppercaseUserInput

選用。指定使用者輸入是否必須為大寫。

類型：布林值

預設：true

useControlMVariable

選用。指定是否要使用 Control-M 規格進行變數取代。

類型：布林值

預設：false

間隙步道批次腳本性質

編碼

選用。批處理項目中使用的編碼（不適用於 groovy）。期望有效的編碼`CP1047IBM930`、`ASCII`、、`UTF-8`...

類型：字串

預設值：ASCII

加普沃克布魯根住宿

經理. 轉碼

選用。對話方塊管理員轉碼對映。可讓您將JICS交易程式碼對應至對話方塊管理員。預期的格式為`trancode1:dialogManager1;trancode2:dialogManager2;`。

類型：字串

預設：null

managers.trancode: OR12:MYDIALOG1

間隙行走 CL 指令性質

命令關閉

選用。要關閉的指令清單，以逗號分隔。允許的值
為PGM_BASICRCVMSG、SNDRCVF、CHGVAR、QCLRDTAQ、、RTVJOBA、ADDLFM、ADDPFM、RCVF、OV
當您要停用或覆寫現有程式時很有用。 PGM_BASIC是專為調試目的而設計的特定 AWS 藍光時代
運行時程序。

類型：字串

預設：null

春天. 數據搜索. 主要. JNDI 名稱

選用。主要的 Java 命名和目錄接口（JNDI）數據源。

類型：字串

默認值：jdbc /主要

zonedMode

選用。編碼或解碼分區資料類型的模式。允許的值
是EBCDIC_STRICT/EBCDIC_MODIFIED/AS400。

類型：字串

預設值：EBCDIC_ STRICT

間隙走道 CL 料道性質

配置. 上下文. 編碼

選用。CL 檔案的編碼。期望有效的編碼CP1047IBM930、ASCII、、UTF-8...

類型：字串

預設值：CP297

CL。 zonedMode

選用。編碼或解碼控制語言 (CL) 指令的模式。允許的值是EBCDIC_STRICT/EBCDIC_MODIFIED/AS400。

類型：字串

預設值：EBCDIC_ STRICT

間隙步道屬JHDB性

小事情程序

選用。要使用的IMS程式清單。使用分號 () 分隔每個參數，並以逗號 (;) 分隔每個交易。,例如：ims.programs: PCP008,PCT008;PCP054,PCT054;PCP066,PCT066;PCP068,PCT068;

類型：字串

預設：null

jhdb。 checkpointPath

選用。如果不jhdb.checkpointPersistence是，none則此參數可讓您設定檢查點持續性路徑 (checkpoint.dat 檔案儲存位置)，登錄中包含的所有檢查點資料都會序列化並備份到提供資料夾中的檔案 (checkpoint.dat) 中。請注意scriptId，stepId此備份只關注檢查點資料 (、資料庫位置和檢查點區域)。

類型：字串

預設值：檔案：。 /設置/

jhdb。 checkpointPersistence

選用。檢查點持久性模式。允許的值是none/add/end。用add於在建立新檢查點並將其新增至登錄時保留檢查點。用end於在伺服器關閉時保留檢查點。任何其他值都會停用持續性。請注意，每次將新的檢查點添加到註冊表時，所有現有的檢查點都將被序列化，並且該文件將被刪除。它不是附加到文件中的現有數據。因此，根據檢查點的數量，它可能會對性能產生一些影響。

類型：字串

預設：none

jhdb. 配置. 上下文. 編碼

選用。JHDB(Java 階層式資料庫) 編碼。需要有效的編碼字串CP1047IBM930、ASCII、、UTF-8...

類型：字串

預設值：CP297

jhdb。 identificationCardData

選用。用於將某些「操作員識別卡數據」硬編碼到CARD參數指定的MID字段中。

類型：字串

預設: ""

日本語

選用。允許您在IMS模擬的情況下強制使用通用邏輯終端機 ID。如果沒有設置 sessionId 則使用。

類型：字串

預設：null

jhdb 元数字. 外部路徑

為 psbs 和 dbds 資料夾指定額外、執行階段特定的根資料夾的組態參數。

類型：字串

預設值：檔案：。 /設置/

> **Note**
>
> 目前，對於部署約束，您必須將 dbds 和 psbs 目錄複製到應用程序的 config 目錄或配置目錄的子目錄中：例如，配置/設置
>
> ```
> config
> |- setup
>    |- dbds
>    |- psbs
> ```
>
> 並在應用程序中設置為 jhdb.yml

```
jhdb.metadata.extrapath: file: ./config/setup/
```

jhdb. 導航. 緩存

選用。的階層式導覽中使用的快取持續時間 (以毫秒為單位) RDBMS。

類型：數字

預設：5000

查詢。 limitJoinUsage

選用。指定是否在RDBMS圖形上使用限制聯結用法參數。

類型：布林值

預設：true

jhdb。 use-db-prefix

選用。指定是否在的階層式導覽中啟用資料庫前置詞RDBMS。

類型：布林值

預設：true

間隙步道屬JICS性

精彩數據。 dataJsonInit位置

選用。分析器從解析準備的 json 文件的位置CSD，並用於初始化 jics 數據庫，

類型：字串

預設: ""

jics.db. dataScriptLocation

選用。initJics.sql 指令碼的位置，由分析器從大型主機剖析CSD匯出時準備。

類型：字串

預設: ""

jics.db. dataTestQuery位置

選用。包含單一 sql 查詢的 sql 指令碼位置，該查詢預期會傳回物件計數 (例如：計算 jics 程式資料表中的記錄數目)。如果計數等於 0，數據庫將使用`jics.db.dataScriptLocation`腳本加載，否則數據庫負載將被跳過。

類型：字串

預設: ""

jics.db. ddlScriptLocation

選用。Jics ddl 指令碼位置。可讓您使用 .sql 命令檔起始 jics 資料庫結構描述。

類型：字串

預設: ""

```
jics.db.ddlScriptLocation: ./jics/sql/jics.sql
```

jics.db. schemaTestQuery位置

選用。應包含唯一查詢的 sql 檔案位置，該查詢會傳回 jics 結構描述 (如果有的話) 中的物件數目。

類型：字串

預設: ""

吉克斯。 runUnitLauncher池. 啟用

選用。指定是否啟動中的執行單元啟動器集區JICS。

類型：布林值

預設：false

吉克斯。 runUnitLauncher游泳池. 大小

選用。執行單位啟動器集區大小JICS。

類型：數字

預設：20

吉克斯。 runUnitLauncher游泳池。 validationInterval

選用性：執行單位啟動器集區的驗證間隔JICS，以毫秒為單位表示。

類型：數字

預設：1000

球隊. 平方. 區域

選用。對 AWS 區域 於 AmazonSQS，用於JICS. 建議將部署的應用程序的性能設置為相同的區域，但它不是強制性的。

類型：字串

默認值：eu-west-1

jics.xa.代理程序/超時

選用。定義負責管理分散式交易的 xa 代理程式完成其作業的持續時間上限。

類型：數字

預設：null

mq. 隊列. 平方米區域

選用。Amazon AWS 區域 SQS MQ 服務。

類型：字串

默認值：eu-west-3

taskExecutor.allowCoreThreadTimeOut

選用。指定是否允許核心執行緒逾時JCIS。即使與非零佇列結合使用，這也可以實現動態增長和縮小 (因為佇列已滿時，最大池大小才會增加)。

類型：布林值

預設：false

taskExecutor.corePoolSize

選用。當終端中的事務通過 groovy 腳本啟動時，將創建一個新的線程。使用此參數可設定核心集區大小。

類型：數字

預設：5

taskExecutor.maxPoolSize

選用。當終端中的事務通過 groovy 腳本啟動時，會創建一個新的 thead。使用此參數可設定集區大小上限 (parallel 執行緒的最大數目)。

類型：數字

預設：10

taskExecutor.queueCapacity

選用。當終端中的事務通過 groovy 腳本啟動時，會創建一個新的 thead。使用此參數可設定佇列大小。(= 達到暫緩交易的最大數目) `taskExecutor.maxPoolSize`

類型：數字

預設：50

間隙行走運行時屬性

cacheMetadata

選用。指定是否快取資料庫中繼資料。

類型：布林值

預設：true

check-groovy-file

選用。指定是否在註冊之前檢查 groovy 文件的內容。

類型：布林值

預設：true

databaseStatistics

選用。指定是否允許SQL建置工具收集和顯示統計資訊。

類型：布林值

預設：false

dateTimeFormat

選用。 dateTimeFormat 說明如何將資料庫日期時間戳記類型溢滿到資料簡化器實體中。允許的值是ISO/EUR/USA/L0CAL

類型：字串

預設值：ISO

dbDateFormat

選用。資料庫目標日期格式。

類型：字串

預設值：年-月-日

dbTimeFormat

選用。資料庫目標時間格式。

類型：字串

默認值：HH：毫米：SS

dbTimestampFormat

選用。資料庫目標時間戳記格式。

類型：字串

默認值：年-月-日高：毫米：SS。 SSSSSS

fetchSize

選用。游標的 fetchSize 值。通過加載/卸載實用程序使用塊獲取數據時使用。

類型：數字

預設：10

forceDisableSQLTrimStringType

選用。指定是否停用所有 sql 字串參數的修剪。

類型：布林值

預設：false

localDateFormat

選用。本地日期格式的列表。使用分隔每種格式 | 。

類型：字串

localTimeFormat

選用。本地時間格式列表。使用分隔每種格式 | 。

類型：字串

localTimestampFormat

選用。本機時間戳記格式清單。使用分隔每種格式 | 。

類型：字串

預設：

pgmDateFormat

選用。程式中使用的日期時間格式。

類型：字串

預設值：年-月-日

pgmTimeFormat

選用。pgm（程序）執行所使用的時間格式。

類型：字串

預設值：HH.mm.ss

pgmTimestampFormat

選用。時間戳記格式。

類型：字串

預設值：年-月-日-高. SSSSSS

間隙行走公用程序屬性

jcl.type

可選。 `.jcl`檔案類型。允許的值是`jcl/vse`。如果檔案對於非 vse jcl 而言為空，IDCAMS公用程式PRINT/REPRO命令會傳回 4。

類型：字串

預設值：mvs

列表目錄. 變量長度前置處理器. 啟用

選用。指定是否為LISTCAT命令啟用可變長度預處理器。

類型：布林值

預設：false

列表目錄. 變量長度前置處理器. 類型

選用。包含在 listcat 檔案中的物件類型 (如果您啟用`listcat.variablelengthpreprocessor.enabled`)。允許的值是rdw/bdw。

類型：字串

預設值：

負載。 batchSize

選用。載入公用程式批次大小。

類型：數字

預設：0

加載. 格式。 dbDate

選用。要使用的載入公用程式資料庫格式。

類型：字串

預設值：年-月-日

加載. 格式。 dbTime

選用。要使用的載入公用程式資料庫時間。

類型：字串

默認值：HH：毫米：SS

加載. 格式。 localDate

選用。要使用的載入公用程式本機日期格式。

類型：字串

預設值：日月年 | 日/月/年 | 年-月-日

加載. 格式。 localTime

選用。要使用的載入公用程式本地時間格式。

類型：字串

默認值：HH：毫米：SS| HH. 毫米

負載。 sqlCodePoint移位

選用。負載公用程式的程式SQL碼點偏移。運行移位字符過程。當你的目標數據庫DB2是 Postgreql 時需要的。

類型：數字

預設：0

sysPunchEncoding

選用。敘述編碼字元集。支援的值為Cp1047/ASCII。

類型：字串

預設值：ASCII

treatLargeNumberAsInteger

選用。指定是否將大數字視為Integer。默認情況BigDecimal下，它們被視為。

類型：布林值

預設：false

卸載。 chunkSize

選用。用於卸載公用程式的區塊大小。

類型：數字

預設：0

卸載。 columnFiller

選用。卸載實用程序列填充物。

類型：字串

預設值：空格

卸載。 fetchSize

選用。可讓您在卸載公用程式中處理游標時調整擷取大小。

類型：數字

預設：0

卸載. 格式.

選用。如果啟unload.useDatabaseConfiguration用，則在卸載公用程式中使用的日期格式。

類型：字串

預設值：月/日/年

卸載. 格式

選用。如果啟unload.useDatabaseConfiguration用，卸載公用程式中使用的時間格式。

類型：字串

預設值：HH.mm.ss

卸載. 格式. 時間戳

選用。如果啟unload.useDatabaseConfiguration用，則在卸載公用程式中使用的時間戳記格式。

類型：字串

預設值：年-月-日-高. SSSSSS

卸載 .nbi。 whenNotNull

選用。當數據庫中的值不為空時，要添加的空字節指示符（nbi）值。

類型：十六進制

預設值：00

卸載 .nbi。 whenNull

選用。當數據庫中的值為空時，要添加的空字節指示器（NBI）值。

類型：十六進制

預設值：6 樓

卸載 .nbi。 writeNullIndicator

選用。指定是否要在卸載輸出檔案中寫出 null 指標。

類型：布林值

預設：false

卸載。 sqlCodePoint移位

選用。卸載實用SQL程序的代碼點移位。運行移位字符過程。當你的目標數據庫DB2是 Postgreql 時需要的。

類型：數字

預設：0

卸載。 useDatabaseConfiguration

選用。指定是否要在卸載公用程式中使用應用程式 main.yml 的日期或時間組態。

類型：布林值

預設：false

卸載。 varCharIs空值

選用。在INFTILB程序中使用此參數，如果設置為`true`則所有不可為空的字段（空格）值返回一個空字符串。

類型：布林值

預設：false

其他屬性

清理. 閾值. 小時

選用。指定何qtemp.dblog時啟用。db 分區存留期（以小時為單位）。

類型：數字

預設：0

博客

選用。是否啟用QTEMP數據庫日誌記錄。

類型：布林值

預設：false

溫度. 長度

選用。QTEMP唯一的 ID 長度。

類型：數字

預設值：9

石英調度程序。 stand-by-if-error

選用。指定工作排程器處於待命模式時是否觸發工作執行。如果為 true，則不會觸發「啟用時」工作執行。

類型：布林值

預設：false

warmUpCache

選用。指定是否要在伺服器啟動時，將所有資料通訊表格資料載入暖機快取。

類型：布林值

預設：false

AWS Mainframe Modernization 應用定義參考

在中 AWS Mainframe Modernization，您可以在應用程式定義JSON檔案中設定已移轉的大型主機應用程式，這是您選擇的執行階段引擎所特有的。應用程式定義包含一般資訊和引擎特定資訊。本主題說明 AWS Blu Age 和 Micro Focus 應用程式定義，並識別所有必要和選用元素。

內容

- 。一般頭部分。
- 定義區段概觀
- AWS 藍光時代的應用程序定義
- AWS 藍光時代定義詳情
  - 監聽程式-必要
  - AWS 藍光時代申請-必填
  - 藍色 SAM-可選
  - AWS 藍色時代消息隊列-可選
  - AWS 藍色時代應用程序存儲EFS配置-可選
- 微焦點應用定義
- 微焦點定義細節
  - 監聽程式-必要
  - 資料集位置-必要
  - Amazon Cognito 可身份驗證和授權處理程序-可選
  - LDAP和活動目錄處理程序-可選
  - Batch 設定-必要
  - CICS設定-必要
  - XA 資源-必要
  - 執行階段設定-選用

。一般頭部分。

每個應用程式定義都會從範本版本和來源位置的一般資訊開始。應用程式定義的目前版本為 2.0。

使用下列結構來指定範本版本和來源位置。

```
"template-version": "2.0",
    "source-locations": [
        {
            "source-id": "s3-source",
            "source-type": "s3",
            "properties": {
                "s3-bucket": "mainframe-deployment-bucket",
                "s3-key-prefix": "v1"
            }
        }
    ]
```

> **Note**
>
> 如果要將 S3 輸入ARN為 S3 桶，可以使用以下語法：
>
> ```
> "template-version": "2.0",
>   "source-locations": [
>       {
>           "source-id": "s3-source",
>           "source-type": "s3",
>           "properties": {
>               "s3-bucket": "arn:aws:s3:::mainframe-deployment-bucket",
>               "s3-key-prefix": "v1"
>           }
>       }
>   ]
> ```

模板版本

必要。指定應用程式定義檔案的版本。請不要變更此值。目前，唯一允許的值是 2.0。以字串指定 `template-version`。

來源位置

指定應用程式在執行階段所需的檔案和其他資源的位置。

來源識別碼

指定位置的名稱。此名稱用於視需要在應用程式定義中參考來源位置JSON。

源類型

指定來源的類型。目前，唯一允許的值是 s3。

屬性

提供來源位置的詳細資訊。每個屬性皆以字串指定。

- `s3-bucket`-必需 指定存放檔案的 Amazon S3 儲存貯體的名稱。
- `s3-key-prefix`-必需 指定儲存檔案之 Amazon S3 儲存貯體中的資料夾名稱。

定義區段概觀

指定應用程式需要執行之服務、設定、資料及其他一般資源的資源定義。更新應用程式定義時，AWS Mainframe Modernization 會比較舊版本`source-locations`和目前應用程式定義JSON檔案中的和`definition`清單來偵測變更。

定義部分是引擎特定的，可能會有所變更。下列各節顯示兩個引擎的引擎特定應用程式定義範例。

AWS 藍光時代的應用程序定義

```
{
 "template-version": "2.0",
 "source-locations": [
        {
            "source-id": "s3-source",
            "source-type": "s3",
            "properties": {
                "s3-bucket": "mainframe-deployment-bucket-aaa",
                "s3-key-prefix": "v1"
            }
        }
    ],
    "definition" : {
        "listeners": [{
            "port": 8194,
            "type": "http"
        }],
        "ba-application": {
            "app-location": "${s3-source}/murachs-v6/"
        },
        "blusam": {
```

```
            "db": {
                "nb-threads": 8,
                "batch-size": 10000,
                "name": "blusam",
                "secret-manager-arn": "arn:aws:secretsmanager:us-
west-2:111122223333:secret:blusam-FfmXLG"
            },
            "redis": {
                "hostname": "blusam.c3geul.ng.0001.usw2.cache.amazonaws.com",
                "port": 6379,
                "useSsl": true,
                "secret-manager-arn": "arn:aws:secretsmanager:us-
west-2:111122223333:secret:bluesamredis-nioefm"
            }
        }
    }
}
```

AWS 藍光時代定義詳情

監聽程式-必要

指定透過 AWS Mainframe Modernization建立的 Elastic Load Balancing 存取應用程式時要使用的連接埠。使用以下結構：

```
"listeners": [{
            "port": 8194,
            "type": "http"
}],
```

port

必要。您可以使用任何可用的端口，除了 0 到 1023 的已知端口。我們建議您使用 8192 到 8199 之間的範圍。確保沒有其他監聽器或應用程序在此端口上運行。

type

必要。目前僅支援 http。

AWS 藍光時代申請-必填

使用下列結構指定引擎拾取應用程式影像檔的位置。

```
"ba-application": {
            "app-location": "${s3-source}/murachs-v6/",
            "files-directory": "/m2/mount/myfolder",
            "enable-jics": <true|false>,
            "shared-app-location": "${s3-source}/shared/"
},
```

應用程式位置

Amazon S3 中存放應用程式映像檔案的特定位置。

文件目錄

選用。批次輸入/輸出檔案的位置。必須是 Amazon EFS 或 Amazon FSx 安裝點設置在環境級別的子文件夾。子文件夾必須由合適的用戶擁有，以供內部 AWS Mainframe Modernization運行的 Blu Age 應用程序使用。為了實現這一目標，將磁碟機連接到 Linux Amazon EC2 執行個體時，3001必須建立具有 ID 的群組101和 ID 的使用者，且所需的資料夾必須由此使用者擁有。例如，通過這種方式，該*testclient*文件夾可以由藍光時代 AWS Mainframe Modernization 管理使用。

```
groupadd -g 101 mygroup
useradd -M -g mygroup -p mypassword -u 3001 myuser
mkdir testclient
chown myuser:mygroup testclient
```

啟用-卡

選用。指定是否啟用JICS。預設為 true。將此項設定為 false 可防止產生JICS資料庫。

shared-app-location

選用。Amazon S3 中存放共用應用程式元素的其他位置。它可以包含相同類型的應用程序結構作為應用程序位置。

藍色 SAM-可選

使用以下結構指定藍光SAM數據庫和 Redis 緩存。

```
"blusam": {
            "db": {
```

```
                "nb-threads": 8,
                "batch-size": 10000,
                "name": "blusam",
                "secret-manager-arn": "arn:aws:secretsmanager:us-
west-2:111122223333:secret:blusam-FfmXLG"
            },
            "redis": {
                "hostname": "blusam.c3geul.ng.0001.usw2.cache.amazonaws.com",
                "port": 6379,
                "useSsl": true,
                "secret-manager-arn": "arn:aws:secretsmanager:us-
west-2:111122223333:secret:bluesamredis-nioefm"
            }
}
```

db

指定與應用程式搭配使用的資料庫屬性。資料庫必須是 Aurora 郵政SQL資料庫。您可以指定下列屬性：

- `nb-threads`-可選。指定 Blu SAM 引擎所依賴的寫入式機制使用多少個專用執行緒。預設值為 8。
- `batch-size`-可選。指定寫入機制用來啟動批次儲存作業的臨界值。臨界值代表將啟動批次儲存作業的已修改記錄數目，以確保已修改的記錄持續存在。觸發程序本身是基於批次大小和一秒的經過時間的組合，以先到達者為準。預設值為 10000。
- `name`-可選。指定資料庫的名稱。
- `secret-manager-arn`-指定包含資料庫登入資ARN料的密碼的 Amazon 資源名稱 ()。如需詳細資訊，請參閱步驟 4：建立和設定 AWS Secrets Manager 資料庫密碼。

Redis

指定應用程式用來將暫存資料儲存在中央位置，以提升效能的 Redis 快取屬性。我們建議您同時加密和密碼保護 Redis 快取。

- `hostname`-指定 Redis 快取的位置。
- `port`-指定 Redis 快取傳送和接收通訊的連接埠 (通常為 6379)。
- `useSsl`-指定 Redis 快取是否加密。如果快取未加密，請設定`useSsl`為 false。
- `secret-manager-arn`-指定包含 Redis 快取密碼的密碼的 Amazon 資源名稱 (ARN)。如果 Redis 快取未受密碼保護，請勿指定。`secret-manager-arn`如需詳細資訊，請參閱步驟 4：建立和設定 AWS Secrets Manager 資料庫密碼。

AWS 藍色時代消息隊列-可選

指定 AWS 藍光時代應用程JMS式的-MQ 連線詳細資料。

```
"message-queues": [
  {
      "product-type": "JMS-MQ",
      "queue-manager": "QMgr1",
      "channel": "mqChannel1",
      "hostname": "mqserver-host1",
      "port": 1414,
      "user-id": "app-user1",
      "secret-manager-arn": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:sample/mq/test-279PTa"
  },
  {
      "product-type": "JMS-MQ",
      "queue-manager": "QMgr2",
      "channel": "mqChannel2",
      "hostname": "mqserver-host2",
      "port": 1412,
      "user-id": "app-user2",
      "secret-manager-arn": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:sample/mq/test-279PTa"
  }
]
```

產品類型

必要。指定產品類型。目前，對於 AWS 藍光時代的應用程序，這只能是「JMS-MQ」。

隊列管理器

必要。指定佇列管理員的名稱。

通道

必要。指定伺服器連線通道的名稱。

hostname

必要。指定訊息佇列伺服器的主機名稱。

port

必要。指定伺服器監聽的監聽器連接埠號碼。

用戶識別碼

選用。指定允許在指定通道上執行訊息佇列作業的使用者帳戶 ID。

secret-manager-arn

選用。指定提供指定使用者密碼的秘密管理員的 Amazon 資源名稱 (ARN)。

AWS 藍色時代應用程序存儲EFS配置-可選

使用下列結構指定應用程式儲EFS存體存取點詳細資料。

```
"ba-application": {
            "file-permission-mask": "UMASK002"
},
"efs-configs": [
           {
            "file-system-id": "fs-01376dfsvfvrsvsr",
            "mount-point": "/m2/mount/efs-ap2",
            "access-point-id": fsap-0eaesefvrefrewgv8"
           }
]
```

file-system-id

必要。存取點套用的EFS檔案系統 ID。模式:「FS-([0-9-F] {8,40}) {1,128} $」

掛載點

必要。應用程式層級檔案系統的掛載點。這必須與環境層級的儲存裝置掛載點不同。

access-point-id

必要。由 Amazon 分配的存取點的 ID EFS。模式:「(0-9-F] {8,40}) {1,128} $」

file-permission-mask

選用。為應用程式處理程序建立的檔案定義檔案建立遮罩。例如，當該值設置為時UMASK006，所有文件都將具有 660 的權限。這意味著只有文件所有者和文件組將具有讀取和寫入訪問權限，而其他用戶沒有任何權限。

> **Note**
>
> 只有在使用應用程式層級EFS儲存時，才會考慮為此欄位設定的值。

> **Note**
>
> 提供 efs config 時，必須在應用程式定義區段中指定檔案目錄。它必須是在應用程式層級設定的 Amazon EFS 掛載點的子資料夾。

微焦點應用定義

下列範例定義區段適用於 Micro Focus 執行階段引擎，且包含必要元素和選用元素。

```
{
 "template-version": "2.0",
 "source-locations": [
        {
            "source-id": "s3-source",
            "source-type": "s3",
            "properties": {
                "s3-bucket": "mainframe-deployment-bucket-aaa",
                "s3-key-prefix": "v1"
            }
        }
    ],
    "definition" : {
        "listeners": [{
            "port": 5101,
            "type": "tn3270"
        }],
        "dataset-location": {
            "db-locations": [{
                "name": "Database1",
                "secret-manager-arn": "arn:aws:secrets:1234:us-east-1:secret:123456"
            }]
        },
        "cognito-auth-handler": {
            "user-pool-id": "cognito-idp.us-west-2.amazonaws.com/us-west-2_rvYFnQIxL",
            "client-id": "58k05jb8grukjjsudm5hhn1v87",
            "identity-pool-id": "us-west-2:64464b12-0bfb-4dea-ab35-5c22c6c245f6"
        },
        "ldap-ad-auth-handler": {
            "ldap-ad-connection-secrets": [LIST OF AD-SECRETS]
        },
        "batch-settings": {
```

```
            "initiators": [{
                "classes": ["A", "B"],
                "description": "initiator...."
            }],
            "jcl-file-location": "${s3-source}/batch/jcl",
            "program-path": "/m2/mount/libs/loadlib:$EFS_MOUNT/emergency/loadlib",
            "system-procedure-libraries":"SYS1.PROCLIB;SYS2.PROCLIB",
            "aliases": [
               {"alias": "FDSSORT", "program": "SORT"},
               {"alias": "MFADRDSU", "program": "ADRDSSU"}
            ]
        },
        "cics-settings": {
            "binary-file-location": "${s3-source}/cics/binaries",
            "csd-file-location": "${s3-source}/cics/def",
            "system-initialization-table": "BNKCICV"
        },
        "xa-resources" : [{
            "name": "XASQL",
            "secret-manager-arn": "arn:aws:secrets:1234:us-east-1:secret:123456",
            "module": "${s3-source}/xa/ESPGSQLXA64.so"
        }],
        "runtime-settings": {
          "environment-variables": {
            "ES_JES_RESTART": "N",
            "EFS_MOUNT": "/m2/mount/efs"
          }
        }
    }
}
```

微焦點定義細節

Micro Focus 應用程式定義檔案的定義區段中的內容會因您移轉的大型主機應用程式在執行階段所需的資源而有所不同。

監聽程式-必要

使用下列結構指定監聽器：

```
"listeners": [{
    "port": 5101,
    "type": "tn3270"
```

```
}],
```

port

對於 TN3270，默認值是 5101。對於其他類型的服務接聽程式，連接埠會有所不同。您可以使用任何可用的端口，除了 0 到 1023 的已知端口。每個接聽程式都應具有獨特的連接埠。監聽器不應共享端口。如需詳細資訊，請參閱 Micro Focus 企業伺服器文件中的監聽程式控制。

type

指定服務接聽程式的類型。如需詳細資訊，請參閱 Micro Focus 企業伺服器說明文件中的監聽器。

資料集位置-必要

使用下列結構指定資料集位置。

```
"dataset-location": {
            "db-locations": [{
                "name": "Database1",
                "secret-manager-arn": "arn:aws:secrets:1234:us-east-1:secret:123456"
            }],
 }
```

DB-位置

指定移轉應用程式所建立之資料集的位置。目前，僅 AWS Mainframe Modernization 支援來自單一資料VSAM庫的資料集。

- `name`-指定包含已移轉應用程式所建立之資料集的資料庫執行個體名稱。
- `secret-manager-arn`-指定包含資料庫登入資ARN料的密碼的 Amazon 資源名稱 ()。

Amazon Cognito 可身份驗證和授權處理程序-可選

AWS Mainframe Modernization 使用 Amazon Cognito 進行已遷移應用程式的身份驗證和授權。使用下列結構指定 Amazon Cognito 身份驗證處理常式。

```
"cognito-auth-handler": {
            "user-pool-id": "cognito-idp.Region.amazonaws.com/Region_rvYFnQIxL",
            "client-id": "58k05jb8grukjjsudm5hhn1v87",
            "identity-pool-id": "Region:64464b12-0bfb-4dea-ab35-5c22c6c245f6"
```

```
}
```

user-pool-id

指定用於驗證已遷移應用程式 AWS Mainframe Modernization 使用者的 Amazon Cognito 使用者集區。用 AWS 區域 於使用者集區應與應用 AWS Mainframe Modernization 程式 AWS 區域 的相符。

客戶端標識

指定已驗證使用者可存取的移轉應用程式。

identity-pool-id

指定 Amazon Cognito 身分集區，經驗證的使用者將使用者集區權杖交換為可讓使用者存取 AWS Mainframe Modernization的登入資料。識別集區的應與應 AWS Mainframe Modernization 用程式 AWS 區域 的相符。 AWS 區域

LDAP和活動目錄處理程序-可選

您可以將應用程式與 Active Directory (AD) 或任何類型的LDAP伺服器整合，讓應用程式的使用者能夠使用其 LDAP /AD 認證進行授權和驗證。

將您的應用程式與 AD 整合

1. 請遵循 Micro Focus 企業伺服器說明文件中的設定企業伺服器安全的作用中目錄中所述的步驟。
2. 為您要與應用程序一起使用的每個 AD/ LDAP 服務器創建 AD/ LDAP 詳細信息的 AWS Secrets Manager 秘密。有關如何建立密碼的資訊，請參閱 AWS Secrets Manager 使用者指南中的建立 Secret Sec AWS rets Manager 碼。對於密碼類型，請選擇其他類型的密碼，並包括下列鍵值配對。

```
{
    "connectionPath"       : "<HOST-ADDRESS>:<PORT>",
    "authorizedId"         : "<USER-FULL-DN>",
    "password"             : "<PASSWORD>",
    "baseDn"               : "<BASE-FULL-DN>",
    "userClassDn"          : "<USER-TYPE>",
    "userContainerDn"      : "<USER-CONTAINER-DN>",
    "groupContainerDn"     : "<GROUP-CONTAINER-DN>",
    "resourceContainerDn"  : "<RESOURCE-CONTAINER-DN>"
}
```

⚠ 安全性建議

- 對於connectionPath， AWS Mainframe Modernization 支援LDAP和 LDAP over SSL (LDAPS) 通訊協定。我們建議使用，LDAPS因為它更安全，並防止憑證出現在網路傳輸中。
- 對於authorizedId和password，我們建議您指定使用者的認證，其權限不超過執行應用程式所需限制最嚴格的唯讀和驗證權限。
- 我們建議您定期輪換 AD/ LDAP 憑證。
- 請勿使用使用者名稱awsuser或建立 AD 使用者mfuser。這兩個用戶名保留供 AWS 使用。

以下是範例。

```
{
    "connectionPath" : "ldaps://msad4.m2.example.people.aws.dev:636",
    "authorizedId" :
 "CN=LDAPUser,OU=Users,OU=msad4,DC=msad4,DC=m2,DC=example,DC=people,DC=aws,DC=dev",
    "password" : "ADPassword",
    "userContainerDn" : "CN=Enterprise Server Users,CN=Micro Focus,CN=Program
 Data,OU=msad4,DC=msad4,DC=m2,DC=example,DC=people,DC=aws,DC=dev",
    "groupContainerDn" : "CN=Enterprise Server Groups,CN=Micro Focus,CN=Program
 Data,OU=msad4,DC=msad4,DC=m2,DC=example,DC=people,DC=aws,DC=dev",
    "resourceContainerDn" : "CN=Enterprise Server Resources,CN=Micro
 Focus,CN=Program Data,OU=msad4,DC=msad4,DC=m2,DC=example,DC=people,DC=aws,DC=dev"
}
```

使用客戶管理的KMS金鑰建立密碼。您必須授 AWS Mainframe Modernization GetSecretValue與密鑰的和DescribeSecret權限，以Decrypt及密KMS鑰的DescribeKey權限。如需詳細資訊，請參閱《 AWS Secrets Manager 使用指南》中的KMS金鑰的權限。

3. 將以下內容添加到您的應用程序定義。

```
"ldap-ad-auth-handler": {
    "ldap-ad-connection-secrets": [LIST OF AD/LDAP SECRETS]
}
```

以下是範例。

```
"ldap-ad-auth-handler": {
    "ldap-ad-connection-secrets": ["arn:aws:secrets:1234:us-east-1:secret:123456"]
}
```

LDAP/AD 驗證處理常式適用於微型焦點 8.0.11 及更新版本。

Batch 設定-必要

使用下列結構指定作為應用程式一部分執行的批次工作所需的詳細資料。

```
"batch-settings": {
            "initiators": [{
                "classes": ["A", "B"],
                "description": "initiator...."
            }],
            "jcl-file-location": "${s3-source}/batch/jcl",
            "program-path": "/m2/mount/libs/loadlib:$EFS_MOUNT/emergency/loadlib",
            "system-procedure-libraries":"SYS1.PROCLIB;SYS2.PROCLIB",
            "aliases": [
               {"alias": "FDSSORT", "program": "SORT"},
               {"alias": "MFADRDSU", "program": "ADRDSSU"}
    ]
}
```

引發劑

指定在遷移的應用程式成功啟動時啟動並繼續執行，直到應用程式停止為止的批次啟動器。您可以為每個初始器定義一個或多個類別。您也可以定義多個初始器。例如：

```
"batch-settings": {
            "initiators": [
                {
                  "classes": ["A", "B"],
                  "description": "initiator...."
                },
                {
                  "classes": ["C", "D"],
                  "description": "initiator...."
```

```
                }

        ],
}
```

如需詳細資訊，請參閱 Micro Focus 企業伺服器說明文件SEP中的若要定義批次啟動器或印表機。

- `classes`-指定啟動器可以執行的工作類別。您最多可以使用 36 個字元。您可以使用下列字元：A-Z 或 0-9。
- `description`-描述初始器的用途。

jcl-file-location

指定移轉應用程式執行的批次 Job 所需的 JCL (工作控制語言) 檔案的位置。

程序路徑

指定當中的程式不在預設位置時執行批次作業所需的路徑。JCL不同的路徑名稱會以冒號 (:) 分隔。

> **Note**
>
> 程式路徑只能是EFS路徑。

system-procedure-libraries

指定要搜尋JCL程序的預設分割資料集。但是，在JCL或透過JCLLIB陳述式中找不到此程序。必須對這些資料集進行分類，且必須使用目錄名稱。並且條目用分號（;）分隔。

別名

定義中使用的公用程式和程式名稱對應JCL至公用程式的實作名稱。 AWS 和第三方批處理實用程序（例如 M2 SFTPWAIT，M2，Syncsort 等）可以選擇性地具有別名，以消除更改. JCL 例如：

- FDSSORTFDSSORT的別名SORT與別FDSICET名 ICETOOL
- ADRDSSUMFADRDSU的別名 ADRDSSU
- 同步排序別名 DMXMFSRT SORT

CICS設定-必要

使用下列結構，指定作為應用程式一部分執行之CICS交易所需的詳細資訊。

```
"cics-settings": {
            "binary-file-location": "${s3-source}/cics/binaries",
            "csd-file-location": "${s3-source}/cics/def",
            "system-initialization-table": "BNKCICV"
}
```

binary-file-location

指定CICS交易程式檔案的位置。

csd-file-location

指定此應用程式的CICS資源定義 (CSD) 檔案的位置。如需詳細CICS資訊，請參閱 Micro Focus 企業伺服器文件中的資源定義。

system-initialization-table

指定移轉的應用程式使用的系統初始化表格 (SIT)。SIT表格名稱最多可包含 8 個字元。您可以使用 A-Z、0-9、$、@ 和 #。如需詳細CICS資訊，請參閱 Micro Focus 企業伺服器文件中的資源定義。

XA 資源-必要

使用下列結構指定應用程式所需的 XA 資源所需的詳細資料。

```
"xa-resources" : [{
            "name": "XASQL",
            "secret-manager-arn": "arn:aws:secrets:1234:us-east-1:secret:123456",
            "module": "${s3-source}/xa/ESPGSQLXA64.so"
}]
```

name

必要。指定 XA 資源的名稱。

secret-manager-arn

指定包含用於連線至資料庫的登入資料的密碼的 Amazon 資源名稱 (ARN)。

模組

指定 RM 交換器模組可執行檔的位置。如需詳細資訊，請參閱 Micro Focus 企業伺服器文件XARs中的規劃和設計。

執行階段設定-選用

使用下列結構指定執行階段設定以管理允許的環境變數所需的詳細資訊。

```
"runtime-settings": {
          "environment-variables": {
            "ES_JES_RESTART": "N",
            "EFS_MOUNT": "/m2/mount/efs"
    }
}
```

環境變量

指定適用於此應用程式執行階段的 Micro Focus 支援的環境變數。

- ES_JES_RESTART是JCL啟用重新啟動處理的 Micro Focus 環境變數。或者，您也可以ES_ALLOC_OVERRIDE將其用作 Micro Focus 環境變數。
- EFS_MOUNT是您的應用程式可用來識別環境EFS掛載位置的自訂環境變數。

您可以存取 Micro Focus 企業伺服器中的所有 Micro Focus 環境變數以取得UNIX指南。

AWS 大型主機現代化資料集定義參考

如果您的應用程式需要多個資料集進行處理，則在 AWS 大型主機現代化主控台中逐一輸入資料集的效率不佳。相反地，我們建議您建立一個JSON檔案來指定每個資料集。不同的資料集類型在中的指定方式不同JSON，雖然許多參數都是共同的。本文件說明匯入不同類型資料集JSON所需的詳細資訊。

> **Note**
>
> 在匯入任何資料集之前，您必須將資料集從大型主機傳輸到。 AWS然後，您必須確定資料集會從大型主機格式轉換為 AWS 可以使用的格式。如有必要，請視需要轉換資料，並將轉換後的資料集存放在 Amazon S3 中。在資料集定義JSON檔案中指定值區和資料夾的名稱。
> 如果您使用的是 Micro Focus 執行階段引擎，則可以使用DFCONV公用程式來轉換資料集。我們將此公用程式納入 Micro Focus 企業開發人員和企業伺服器映像中。如需詳細資訊，請參閱 Micro Focus 企業開發人員文件中的 DFCONVBatch 檔案轉換。

主題

- 一般屬性

- 資料集請求格式範例 VSAM
- GDG基礎的範例資料集請求格式
- PS 或GDG層代的範例資料集請求格式
- PO 的資料集請求格式範例

一般屬性

數個參數是所有資料集通用的。這些參數涵蓋下列區域：

- 有關資料集的資訊 (datasetNamedatasetOrg、、recordLength、encoding)
- 有關匯入來源位置的資訊；也就是資料集的來源位置。這不是大型主機上的位置。這是您上傳資料集 (externalLocation) 之 Amazon S3 位置的路徑。
- 有關匯入目標位置的資訊；亦即資料集的目標位置。此位置可能是資料庫或檔案系統，視您的執行階段引擎而定。 (storageType和relativePath)。
- 有關資料集類型 (特定資料集類型、格式、編碼等) 的資訊。

每個資料集定義都有相同的JSON結構。下列範例JSON顯示所有這些常用參數。

```
{
    "dataSet": {
        "storageType": "Database",
        "datasetName": "MFI01V.MFIDEMO.BNKACC",
        "relativePath": "DATA",
        "datasetOrg": {
            "type": {
                type-specific properties
                    ...
            },
        },
    },
}
```

下列屬性適用於所有資料集。

storageType

必要。套用至目標位置。指定資料集是儲存在資料庫還是檔案系統中。可能的值為 Database 或 FileSystem。

- AWS 藍光時代運行時引擎：不支持文件系統。您必須使用資料庫。
- Micro Focus 運行時引擎：數據庫和文件系統都支持。您可以將 Amazon Relational Database Service 或 Amazon Aurora 用於數據庫，也可以使用 Amazon Elastic File System 或 Amazon FSx 的 Lustre 文件系統文件系統。

datasetName

(必要) 指定資料集在大型主機上顯示的完整名稱。

relativePath

(必要) 套用至目標位置。指定資料集在資料庫或檔案系統中的相對位置。

datasetOrg

(必要) 指定資料集的類型。可能值為 vsam、gdg、ps、po 或 unknown。

- AWS 藍光時代運行時引擎：僅支持VSAM類型數據集。
- 微焦點運行時引擎：VSAMGDG，，PS，PO 或未知類型的數據集支持。

> **Note**
>
> 如果您的應用程序需要的文件不是COBOL數據文件，而是PDF或其他二進制文件，則可以按如下方式指定它們：
>
> ```
> "datasetOrg": {
>             "type": PS {
>                 "format": U
>             },
> ```

資料集請求格式範例 VSAM

- AWS 藍色時代運行時引擎：支持。
- 微焦點運行時引擎：支持。

如果您要匯入VSAM資料集，請指定vsam為datasetOrg。您JSON應該類似於以下示例：

```
{
    "storageType": "Database",
```

```
    "datasetName": "AWS.M2.VSAM.KSDS",
    "relativePath": "DATA",
    "datasetOrg": {
        "vsam": {

            "encoding": "A",
            "format": "KS",
            "primaryKey": {
                "length": 11,
                "offset": 0
            }
        }
    },
    "recordLength": {
        "min": 300,
        "max": 300
    }
},
"externalLocation": {
    "s3Location": "s3://$M2_DATA_STORE/catalog/data/AWS.M2.VSAM.KSDS.DAT"
}
```

VSAM資料集支援下列性質。

編碼

(必要) 指定資料集的字元集編碼。可能的值包括 ASCII EBCDIC (A)、(E) 和未知 (?)。

格式

(必要) 指定VSAM資料集類型和記錄格式。

- AWS 藍色時代運行時引擎：可能的值是 ESDSKSDS（ESKS），（ ）和RRDS（RR）。記錄格式可以是固定的或可變的。
- 微焦點運行時引擎：可能的值是 ESDSKSDS（ESKS），（ ）和RRDS（RR）。該定VSAM義包括記錄格式，因此您不需要單獨指定它。

primaryKey

(必要) 僅套用至VSAMKSDS資料集。指定主鍵。由主鍵名稱、鍵位移和鍵長度組成。name是選擇性的；offset且length是必要的。

recordLength

(必要) 指定記錄的長度。對於固定長度的記錄格式，這些值必須相符。

- AWS 藍光 Age 運行時引擎：for VSAM ESDSKSDS，和RRDS，`min`是可選的，`max`是必需的。
- 微焦點運行時引擎：`min`和`max`是必需的。

externalLocation

(必要) 指定來源位置：亦即您上傳資料集的 Amazon S3 儲存貯體。

藍光時代引擎特定性質

AWS 藍光時代運行時引擎支持VSAM數據集的壓縮。下列範例顯示如何在中指定此屬性JSON。

```
{
    common properties
        ...
        "datasetOrg": {
            "vsam": {
                common properties
                ...
                "compressed": boolean,
                common properties
                ...
            }
        }
}
```

指定壓縮屬性，如下所示：

compression

(選擇性) 指定此資料集的索引是否儲存為壓縮值。如果您有較大的資料集 (通常 > 100 Mb)，請考慮將此旗標設定為`true`。

GDG基礎的範例資料集請求格式

- AWS 藍色時代運行時引擎：不支持。
- 微焦點運行時引擎：支持。

如果要匯入GDG基準資料集，請指定`gdg`為`datasetOrg`。您JSON應該類似於以下示例：

```
{
```

```
    "storageType": "Database",
    "datasetName": "AWS.M2.GDG",
    "relativePath": "DATA",
    "datasetOrg": {
        "gdg": {
            "limit": "3",
            "rollDisposition": "Scratch and No Empty"
        }
    }
}
```

GDG基礎資料集支援下列屬性。

limit

(必要) 指定作用中層代或偏差的數目。對於GDG基本叢集，最大值為 255。

rollDisposition

(選擇性) 指定達到或超過最大值時如何處理產生資料集。可能值為 `No Scratch and No Empty`、`Scratch and No Empty`、`Scratch and Empty`、 或 `No Scratch and Empty`。預設值為 `Scratch and No Empty`。

PS 或GDG層代的範例資料集請求格式

- AWS 藍色時代運行時引擎：不支持。
- 微焦點運行時引擎：支持。

如果您要匯入 PS 或GDG層代資料集，請指定`ps`為`datasetOrg`。您JSON應該類似於以下示例：

```
{
    "storageType": "Database",
    "datasetName": "AWS.M2.PS.FB",
    "relativePath": "DATA",
    "datasetOrg": {
        "ps": {
            "format": "FB",
            "encoding": "A"
        }
    },
    "recordLength": {
```

```
        "min": 300,
        "max": 300
    }
},
"externalLocation": {
    "s3Location": "s3://$M2_DATA_STORE/catalog/data/AWS.M2.PS.LSEQ"
}
}
```

PS 或GDG層代資料集支援下列內容。

格式

(必要) 指定資料集記錄的格式。可能的值
為FFA、FB、FBA、FBM、FBS、FM、FS、LSEQ、U、V、VA、VB、、VBA、VBM、VBS、VM、
和VS。

編碼

(必要) 指定資料集的字元集編碼。可能的值為 ASCII (A)、EBCDIC (E) 和未知 (?)

recordLength

(必要) 指定記錄的長度。您必須同時指定記錄的最小長度 (minmax) 和最大 () 長度。對於固定長度的記錄格式，這些值必須相符。

externalLocation

(必要) 指定來源位置：亦即您上傳資料集的 Amazon S3 儲存貯體。

PO 的資料集請求格式範例

如果您要匯入 PO 資料集，請指定po為datasetOrg. 您JSON應該類似於以下示例：

```
{
    "storageType": "Database",
    "datasetName": "AWS.M2.PO.PROC",
    "relativePath": "DATA",
    "datasetOrg": {
        "po": {
            "format": "LSEQ",
            "encoding": "A",
            "memberFileExtensions": ["PRC"]
```

```
        }
    },
    "recordLength": {
        "min": 80,
        "max": 80
    }
},
"externalLocation": {
    "s3Location": "s3://$M2_DATA_STORE/source/proc/"
}
}
```

PO 資料集支援下列內容。

格式

(必要) 指定資料集記錄的格式。可能的值
為FFA、FB、FBA、FBM、FBS、FM、FS、LSEQ、U、V、VA、VB、、VBA、VBM、VBS、VM、
和VS。

編碼

(必要) 指定資料集的字元集編碼。可能的值包括 ASCII EBCDIC (A)、(E) 和未知 (?)。

memberFileExtensions

(必要) 指定包含一或多個副檔名的陣列，可讓您指定要包含哪些檔案作為PDS成員。

recordLength

(選擇性) 指定記錄的長度。記錄的最小（`min`）和最大（`max`）長度都是可選的。對於固定長度的記錄格式，這些值必須相符。

externalLocation

(必要) 指定來源位置：亦即您上傳資料集的 Amazon S3 儲存貯體。

> **Note**
>
> Micro Focus 執行階段引擎目前的實作會將PDS項目新增為動態資料集。

AWS 大型主機現代化中的受管理執行階段環境

如果您是 AWS 大型主機現代化的新手，請參閱下列主題以開始使用：

- 什麼是 AWS 大型主機現代化？
- 設定 AWS 大型主機現代化
- 開始使用 AWS 大型主機現代化
- 教程：為 AWS 藍光時代設置託管運行時
- 教學課程：設定 Micro Focus 的受管理執行階段

AWS 大型主機現代化中的執行階段環境是 AWS 運算資源、執行階段引擎和您指定的組態詳細資料的具名組合。執行階段環境會裝載一或多個應用程式。大型主機現代 AWS 化中的應用程式包含移轉的大型主機工作負載。您可以為您建立的環境選擇執行階段引擎。如果您使用的是自動重構模式，請選擇「 AWS 藍光時代」；如果您使用的是重構模式，請選擇 Micro Focus。您也可以選擇適合應用程式的運算資源數量，並選擇性地將儲存裝置附加至執行階段環境。 AWS 大型主機現代化可為您啟用 Amazon CloudWatch 指標和記錄，以便您監控執行階段環境。

主題

- 建立 AWS 大型主機現代化執行階段環境
- 更新 AWS 大型主機現代化執行階段環境
- 停止 AWS 大型主機現代化執行階段環境
- 重新啟動 AWS 大型主機現代化執行階段環境
- 刪除 AWS 大型主機現代化執行階段環境

建立 AWS 大型主機現代化執行階段環境

使用 AWS 大型主機現代化主控台建立大型主機現代 AWS 化環境。

這些指示假設您已完成中的步驟設定 AWS 大型主機現代化。

建立執行階段環境

建立執行階段環境

1. 在開啟大 AWS 型主機現代化主控台。https://console.aws.amazon.com/m2/

2. 在選取 AWS 區域 器中，選擇您要建立環境的「區域」。
3. 在 [環境] 頁面上，選擇 [建立環境]。
4. 在 [指定基本資訊] 頁面上，提供下列資訊：
   a. 在「名稱和描述」區段中，輸入環境的名稱。
   b. (選用)。在「環境描述」欄位中，輸入環境的描述。此描述可協助您和其他使用者識別執行階段環境的用途。
   c. 在「引擎選項」區段中，選擇「藍光時代」進行自動重構，或選擇 Micro Focus 進行重構。
   d. 選擇所選引擎的版本。
   e. (選用)。在「標籤」區段中，選擇「新增標籤」，將一或多個環境標籤新增至您的環境。環境標籤是一種自訂屬性標籤，可協助您組織和管理 AWS 資源。
   f. 選擇 Next (下一步)。
5. 在 [指定組態] 頁面上，提供下列資訊：
   a. 在「使用狀態」段落中，選擇獨立執行環境或高可用性叢集。

      可用性模式會決定應用程式執行時的可用性。獨立對於開發目的很好。高可用性適用於必須始終可用的應用程式。
   b. 在資源中，選擇執行個體類型和所需的容量。

      這些資源是將託管您執行時間環境的 AWS 大型主機現代化受管 Amazon EC2 執行個體。獨立執行階段環境提供兩種執行個體類型選擇，而且只允許一個執行個 高可用性執行階段環境為執行個體類型提供兩種選擇，最多允許兩個執行個體。

      如需詳細資訊，請參閱 Amazon EC2 執行個體類型，並聯絡大型 AWS 主機專家以取得指導。
6. 在「安全性與網路」區段中，執行下列動作：
   a. 如果您希望應用程式可公開存取，請選擇 [允許部署到此環境的應用程式可公開存取]。
   b. 選擇虛擬私有雲 (VPC)。
   c. 如果您使用的是高可用性模式，請選擇兩個或多個子網路。如果您使用 AWS Blu Age 引擎的獨立模式，請選擇兩個或更多的子網路。如果您使用 Micro Focus 引擎的獨立模式，您可以指定一個子網路。
   d. 為您選取的安全性群組選擇一VPC個安全性群組。

> **Note**
>
> AWS 大型主機現代化會建立 Network Load Balancer，讓您將連線分配到執行階段環境。請確定您的安全群組輸入規則允許從 IP 位址存取您在應用程式定義`listener`屬性中指定的連接埠。如需詳細資訊，請參閱網路負載平衡器使用者指南中的註冊目標。

e. 如果您要使用受管理的客戶，請在KMS金鑰欄位中選擇「自訂加密設定」 AWS KMS key。如需詳細資訊，請參閱AWS 大型主機現代化服務的靜態資料加密。

> **Note**
>
> 根據預設， AWS 大型主機現代化會使用您擁有和管理的 AWS 大型主機現代化 AWS KMS key 來加密您的資料。但是，您可以選擇使用受管理的客戶 AWS KMS key。

f. (選擇性) 選擇 AWS KMS key 依名稱或 Amazon 資源名稱 (ARN)。或者，選擇 [建立] AWS KMS key以前往 AWS KMS 主控台並建立新 AWS KMS key的。

g. 選擇 Next (下一步)。

7. (選擇性) 在 [連接儲存] 頁面上，選擇一或多個 Amazon EFS 或 Amazon FSx 檔案系統，然後選擇 [下一步]。

8. 在「維護時段」 段落中，選擇要將暫止變更套用至環境的時機。

   - 如果您選擇 [無偏好設定]， AWS 大型主機現代化會為您選擇最佳化的維護時段。
   - 如果您要指定特定的維護時段，請選擇「選取新的維護時段」。然後選擇星期幾、開始時間和維護時段的持續時間。

   如需維護時段的詳細資訊，請參閱AWS 大型主機現代化維護視窗。

   選擇 Next (下一步)。

9. 在 [檢閱並建立] 頁面上，檢閱您輸入的資訊，然後選擇 [建立環境]。

更新 AWS 大型主機現代化執行階段環境

使用 AWS 大型主機現代化主控台來更新大型主機現代化執 AWS 行階段環境。您可以更新執行階段引擎的次要版本，或更新裝載執行階段環境的執行個體類型。您可以選擇要立即套用更新，還是要在偏好的維護期間套用更新。

這些說明假設您已完成設定 AWS 大型主機現代化中的步驟。

更新執行階段環境

更新執行階段環境

1. 在開啟大 AWS 型主機現代化主控台。https://console.aws.amazon.com/m2/
2. 在選取 AWS 區域 器中，選擇建立您要更新之環境的「區域」。
3. 在 [環境] 頁面上，選擇您要更新的環境。
4. 在環境的詳細資訊頁面上，選擇「動作」，然後選擇 「編輯環境」。
5. 進行下列任何變更：
   - 在 [引擎選項] 區段中，選擇您想要的引擎版本。
   - 在 [資源] 區段中，選擇您要的執行個體類型。
   - 在「維護時段」 區段中，選擇您想要的日期、時間和持續時間。

   > **Note**
   >
   > 您可以選擇在維護期間套用的唯一變更是引擎版本的變更。您必須立即套用所有其他變更。
6. 選擇 Next (下一步)。
7. 在套用這些變更的時機中，選擇立即或在下一個維護時段期間。然後選擇更新環境。

如果您選擇「立即」，則會在環境完成更新時看到訊息。

AWS 大型主機現代化維護視窗

每個執行階段環境都有每週兩小時的維護時段。在此期間會套用任何系統變更。維護時段是您控制何時進行修改以及進行軟體和安全性修補的機會。如果維護事件排定在指定週內，則維護事件會在該兩小時

的維護時段內開始。大多數維護事件也會在兩小時的維護時段內完成，雖然較大的維護事件可能需要幾個小時以上才能完成。

從每個區域 8 小時的時間段中隨機選擇兩小時的維護時段。如果您在建立執行階段環境時未指定維護時段， AWS 大型主機現代化會在一週中隨機選取的日期指派 2 小時的維護時段。

AWS 在套用維護時，大型主機現代化會消耗環境執行個體中的部分資源。在維護期間，您可能會觀察到對效能或應用程式中斷的影響最小。

下表顯示為每個「區域」指定維護時段時間時的預設時間區塊。

區域名稱	區域	時間區塊
美國東部 (維吉尼亞北部)	us-east-1	凌晨三時至十一時 UTC
美國西部 (奧勒岡)	us-west-2	上午 6 時至下午 2 時 UTC
亞太區域 (孟買)	ap-south-1	上午 6 時至下午 2 時 UTC
亞太區域 (新加坡)	ap-southeast-1	下午 2 時至晚上 10 時 UTC
亞太區域 (悉尼)	ap-southeast-2	12 時至晚上 8 時 UTC
亞太區域 (東京)	ap-northeast-1	下午一時至下午九時 UTC
加拿大 (中部)	ca-central-1	凌晨三時至十一時 UTC
歐洲 (法蘭克福)	eu-central-1	下午 9 時至 5 時 UTC
歐洲 (愛爾蘭)	eu-west-1	晚上 10 時至 6 時 UTC
歐洲 (倫敦)	eu-west-2	晚上 10 時至 6 時 UTC
歐洲 (巴黎)	eu-west-3	晚上七時五十九分 UTC
南美洲 (聖保羅)	sa-east-1	上午十二時至八時 UTC

停止 AWS 大型主機現代化執行階段環境

使用 AWS 大型主機現代化主控台停止大型主機現代化執 AWS 行階段環境。當您停止環境時，會保留目前的應用程式部署，而且在重新啟動環境之前，您不需要支付環境費用。

這些說明假設您已完成設定 AWS 大型主機現代化中的步驟。

停止執行階段環境

如果您需要停止 AWS 大型主機現代化執行階段環境，請遵循與更新環境區段類似的步驟。

使用 AWS 大型主機現代化主控台停止大型主機現代化執 AWS 行階段環境。當您停止環境時，會保留目前的應用程式部署，而且在重新啟動環境之前，您不需要支付環境費用。

> **Note**
>
> 您必須在停止環境之前停止所有應用程式。

若要停止執行階段環境

1. 在開啟大 AWS 型主機現代化主控台。https://console.aws.amazon.com/m2/
2. 在選取 AWS 區域 器中，選擇建立您要停止之環境的「區域」。
3. 在「環境」頁面上，選擇您要停止的環境。
4. 在環境的詳細資訊頁面上，選擇「動作」，然後選擇 「編輯環境」。
5. 在 [編輯環境] 頁面上，尋找 [資源] 區段，並將所需的容量更新為零。

   > **Note**
   >
   > 若要停止環境，您只能選擇立即停止。

6. 選擇 Next (下一步)。
7. 在套用這些變更的時機中，選擇立即。然後選擇更新環境。

更新環境容量時，您會看到一則訊息。

重新啟動 AWS 大型主機現代化執行階段環境

使用 AWS 大型主機現代化主控台重新啟動大型主機現代化執 AWS 行階段環境。當您重新啟動執行階段環境時，將會繼續計費環境。

重新啟動執行環境

若要重新啟動 AWS 大型主機現代化執行階段環境，請遵循與停止環境區段類似的步驟。

重新啟動執行階段環境

1. 在開啟大 AWS 型主機現代化主控台。https://console.aws.amazon.com/m2/
2. 在選取 AWS 區域 器中，選擇您要重新啟動之環境建立的區域。
3. 在 [環境] 頁面上，選擇您要重新啟動的環境。
4. 在環境的詳細資訊頁面上，選擇「動作」，然後選擇 「編輯環境」。

> **Note**
>
> 獨立環境所需的容量只能更新為 1。若要重新啟動執行階段環境，您只能選擇立即重新啟動。

5. 在 [編輯環境] 頁面上，尋找 [資源] 區段，然後將所需容量從零更新為所需容量。
6. 選擇 Next (下一步)。
7. 在套用這些變更的時機中，選擇立即。然後選擇更新環境。

當環境容量更新並重新啟動環境時，您會看到一則訊息。

刪除 AWS 大型主機現代化執行階段環境

使用 AWS 大型主機現代化主控台刪除大型主機現代化執 AWS 行階段環境。

這些說明假設您已完成設定 AWS 大型主機現代化中的步驟。

刪除執行階段環境

如果您需要刪除 AWS 大型主機現代化執行階段環境，請務必先從環境中刪除任何已部署的應用程式。您無法刪除部署應用程式的執行階段環境。

刪除環境

1. 在開啟大 AWS 型主機現代化主控台。https://console.aws.amazon.com/m2/
2. 在選取 AWS 區域 器中，選擇建立您要刪除之環境的「區域」。
3. 在 [環境] 頁面上，選擇您要刪除的環境，然後選擇 [動作] 和 [刪除環境]。
4. 在 [刪除環境] 視窗中，輸入`delete`以確認您要刪除執行階段環境，然後選擇 [刪除]。

AWS 大型主機現代化中的應用程式測試

AWS 大型主機現代化應用程式測試可為您的移轉專案提供自動化功能等效測試。 AWS 大型主機現代化應用程式測試運用雲端的彈性，加速移轉專案的速度。您可以根據需要在任意數量的 parallel 環境上執行獨立的測試套件，從而減少測試時間表。應用程式測試的主要優點包括測試加速和敏捷性、高度的測試重複性、內建的可擴充性和彈性、大規模自動化、成本效益，以及與 AWS CloudFormation 建立目標測試環境的無縫整合。

主題

- 什麼是 AWS 大型主機現代化應用程式測試？
- AWS 大型主機現代化應用程式測試概念
- AWS 大型主機現代化應用程式測試先決條件
- 應用程式測試主控台
- 教學課程：在 AWS 大型主機現代化應用程式測試中設定 CardDemo 範例應用程式
- 教學課程：使用 Amazon 上部署的 AWS Blu AWS Age，在大型主機現代化應用 CardDemo 程式測試中重播和比較 EC2
- AWS 大型主機現代化應用程式測試支援的資料集程式碼頁
- AWS 大型主機現代化應用程式測試中的資料保護

什麼是 AWS 大型主機現代化應用程式測試？

測試會大幅影響移轉專案。它最多可以消耗 70% 的遷移、現代化或增強專案時間和精力。 AWS 應用程式測試是 AWS 大型主機現代化的一項功能，可為您移轉的應用程式提供自動化功能等效測試。功能等效測試可協助您驗證上的應用程式 AWS 雲端 是否等同於大型主機上的應用程式。 AWS 應用程式測試會自動比較大型主機與之間的資料集、資料庫記錄和線上 3270 畫面的變更。 AWS此外，「應用程式測試」允許重複測試，因此您可以在更新目標架構、解決問題並朝著完全移轉的應用程式進展時多次執行測試案例。移轉之後，您可以繼續使用應用程式測試進行回歸測試，以確定執行階段引擎或其他元件的更新不會造成回歸。應用程序測試具有成本效益：使用用戶提供的 CloudFormation 模板創建目標測試環境，利用基礎架構即代碼（IAC）概念。應用程式測試使用雲端的彈性加速移轉專案。您可以視需要在任意數量的 parallel 環境上執行獨立的測試套件，以減少測試時間表。

主題

- 您是第一次使用應用程式測試嗎？

- 應用程序測試的好處
- 與整合 AWS CloudFormation
- 應用程式測試的運作
- 相關服務
- 訪問應用測試
- 應用程式測試的定價

您是第一次使用應用程式測試嗎？

如果您是應用程式測試的初次使用者，建議您先閱讀下列章節：

- 應用程式測試概
- 教學：在 CardDemo 應用程式測試中設定應用程式
- the section called “教程：使用 AWS 藍光時代重播和比較 CardDemo”

應用程序測試的好處

應用程式測試提供數個好處，可協助您完成移轉程序：

- 測試加速度，敏捷性和靈活性。
- 「在大型機上錄製一次，在測試概念中多次重播AWS」。
- IaC 通過用戶提供 CloudFormation 的模板創建目標環境。
- 高度的測試重複性。
- 專為雲端打造，考量擴充性和彈性。
- 具有高度自動化的大規模測試。
- 成本效益。

與整合 AWS CloudFormation

應用程序測試使用基礎設施作為代碼 AWS CloudFormation. 此設計選擇可簡化並改善您的測試體驗。AWS CloudFormation 為您提供自主性和獨立性，為您的需求定義更好的基礎架構。您可以單獨選取或定義許多參數 (執行個體大小、RDS執行個體、最佳安全性群組)。您可以新增資源，例如 Amazon SQS 佇列，讓應用程式在測試條件下正常運作所需的資源。

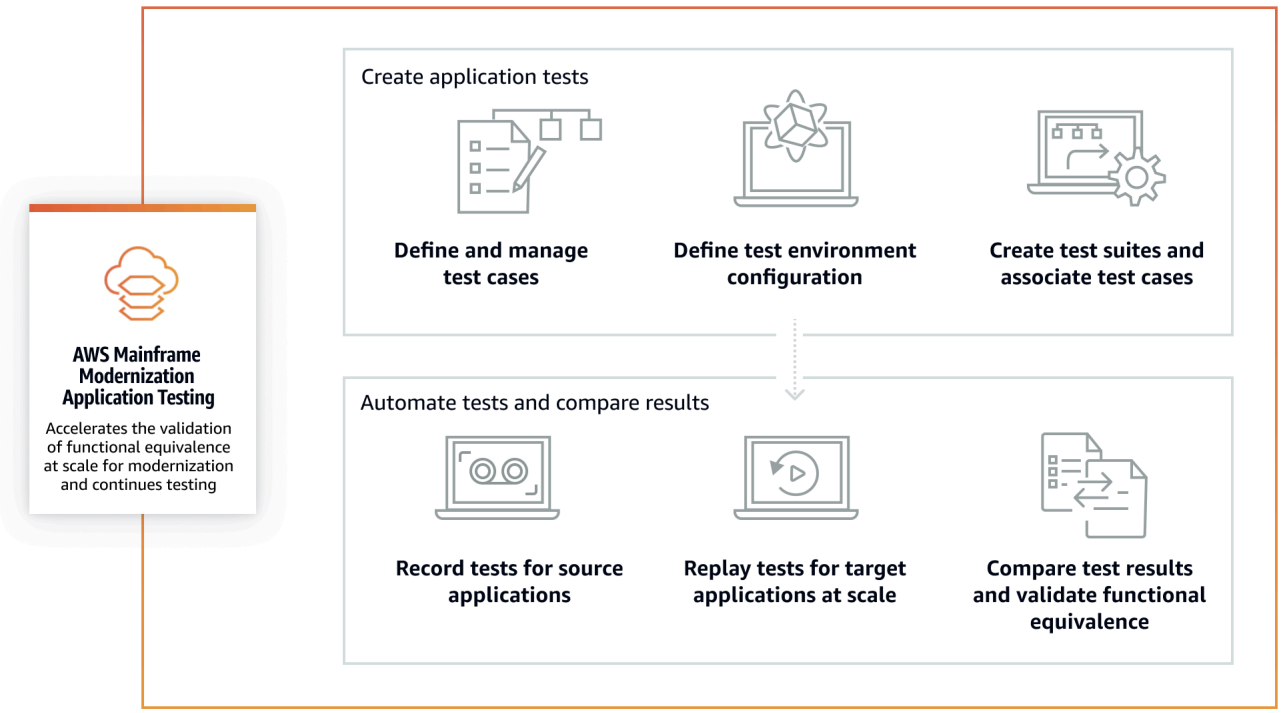
在提供下載的 AWS CloudFormation 模板中，您會注意到一些常見功能：

- 應用程式測試會建立完全隔離的堆疊，包括 AWS 大型主機現代化執行階段環境和應用程式，並具有自己的網路和安全性定義。此隔離堆疊提供彈性，因為相同的其他參與者 AWS 帳戶 不會干擾測試活動。它還可以避免系統操作員修改默認VPC或安全組，這可能會導致測試活動失敗的情況。
- 安全組還允許您控制對測試中使用的資源的外部訪問。例如，資料庫可能包含機密資料。
- 完全隔離可防止其他共享內容VPC的參與者窺探流量。
- 它增強了性能。例如，範本建立的 AWS 大型主機現代化應用程式與其 Amazon RDS 資料庫之間的通訊發生在單獨的網路 (私有網路VPC) 上，以避免其他參與者減慢流量。

我們建議您也在建立的 AWS CloudFormation 範本中實作這些功能。

應用程式測試的運作

下圖是應用程序測試是如何工作的概述。



- 您可以 AWS 使檔案傳輸用您偏好的大型主機資料傳輸工具，將輸入資料從來源傳輸到。
- 您可以在來源和目標上執行相同的商務邏輯。
- 應用程序測試會自動比較來源和目標的輸出數據（數據集，關係數據庫更改，在線 3270 屏幕和用戶交互）。在大型主機上執行測試案例之後，您擷取輸出資料並將其傳輸到 AWS，然後在目標上重播

測試案例。應用程序測試會自動從測試運行的輸出數據 AWS 與來自源的輸出數據進行比較。您可以一目了然地看到哪些記錄是相同的，等同的，不同的，或缺少。此外，您可以定義對等規則，以便將不相同但具有相同商業意義的記錄理解為對等。

您在「應用程式測試」中遵循的工作流程包含下列步驟：

1. 創建測試用例：測試用例是測試操作的最小單元。當您建立測試案例時，您也會識別要比較的資料類型，這些資料類型最能代表來源與目標之間的功能對等。
2. 定義測試環境配置：通過指定 AWS CloudFormation 模板和其他屬性來指定您的環境配置。
3. 創建測試套件：測試套件是測試用例的集合。
4. 在來源上傳資料集並在目標上重新顯示：擷取大型主機上的輸入和輸出資料集，並將其上傳至。AWS然後在上重播測試方案 AWS。
5. 比較來源和目標資料集：「應用程式測試」會自動比較來源和目標的輸出資料集，讓您一目了然地看到哪些是正確的，哪些不正確。

測試場景的最後一個動作和整個過程的目標是識別源和目標測試運行之間的差異。應用程序測試比較源版本和目標版本的測試運行期間在所有交互通道上捕獲的數據。它還比較了相關數據的最終狀態（如測試用例中所定義）。

相關服務

應用程式測試是 AWS 大型主機現代化的一項功能。它還使用基礎設施作為代碼， AWS CloudFormation 以確保測試的可重複性，自動化和成本效益。如需詳細資訊，請參閱：

- AWS 大型主機現代化
- AWS CloudFormation

訪問應用測試

您可以在左側導覽窗格中選擇「應用程式測試」，https://console.aws.amazon.com/apptest/或從「 AWS 大型主機現代化」主控台存取「應用程式測試」主控台。

應用程式測試的定價

您可以在AWS大型主機現代化定價中找到應用程式測試的定價。

AWS 大型主機現代化應用程式測試概念

AWS 應用程序測試使用的術語，其他測試服務或軟件包可能與略有不同的含義使用。下列各節說明 AWS 大型主機現代化應用程式測試如何使用此術語。

主題

- 測試用例
- 測試套件
- 測試環境配置
- 上傳
- 重新播放
- Compare
- 資料庫比較
- 資料集比較
- 比較狀態
- 等價規則
- 最終狀態資料集比較
- 狀態進度資料庫比較
- 功能等價 (FE)
- 在線屏幕比較
- 重播資料
- 參考資料
- 上傳、重播和比較
- 差異
- 等值
- 來源應用
- 目標應用

測試用例

測試用例是測試工作流程中單獨最原子的行動單元。通常情況下，測試用例用於表示修改數據的業務邏輯的獨立單元。將為每個測試用例進行比較。測試用例被添加到測試套件。測試用例包含有關測試用

例修改的數據工件（數據集，數據庫）以及有關測試用例執行期間觸發的業務功能的元數據：批處理作業，3270 交互式對話框等。例如，資料集的名稱和字碼頁。

輸入數據 → 測試用例 → 輸出數據

測試用例可以是在線或批處理類型：

- 在線 3270 屏幕測試用例是用戶執行交互式屏幕對話框（3270）以讀取，修改或生成新的業務數據（數據庫和/或數據集記錄）的測試用例。
- Batch 測試用例是需要提交批處理以讀取，處理和修改或生成新的業務數據（數據集和/或數據庫記錄）的測試用例。

測試套件

測試套件具有按順序運行的測試用例的集合，一個接一個。重播是在測試套件級別完成的。重播測試套件時，測試套件中的所有測試用例都在目標測試環境上運行。如果在比較參考和重播測試工件之後存在差異，則差異將顯示在測試用例級別。

例如，測試套件 A：

測試案例 1、測試案例 2、測試案例 3 等。

測試環境配置

測試環境配置允許您設置初始一組數據和配置參數（或資源） CloudFormation ，以使測試運行可重複。

上傳

上傳是在測試套件級別完成的。在上傳期間，您必須提供 Amazon S3 位置，其中包含來源大型主機中要與之比較的成品、資料集和關聯式資料庫CDC日誌。這些資料將被視為來源大型主機的參考資料。在重新執行期間，系統會將產生的重新執行資料與上傳的參照資料進行比較，以確保應用程式等同

重新播放

重播是在測試套件級別完成的。在重新執行期間， AWS 大型主機現代化應用程式測試會使用指CloudFormation令碼建立目標測試環境並執行應用程式。系統會擷取重新執行期間修改的資料集和資料庫記錄，並與大型主機的參考資料進行比較。一般而言，您會在大型主機上上傳一次，然後重播多次，直到達到功能等效為止。

Compare

重新執行成功完成後，系統會自動進行比較。在比較期間，您在上傳階段上傳及擷取的參照資料會與重新執行階段期間產生的重新執行資料進行比較。數據集，數據庫記錄和在線屏幕單獨的測試用例級別進行比較。

資料庫比較

應用程序測試比較源和目標應用程序之間的數據庫記錄的變化時，採用狀態進度匹配功能。狀態進度比對會比較每個個別執行INSERT、UPDATE和DELETE陳述式中的差異，這與在程序結束時比較資料表資料列不同。狀態進度匹配比替代方案更有效，通過僅比較變更的數據並檢測交易流程中的自我糾正錯誤，從而提供更快，更準確的比較。通過使用CDC（更改數據捕獲）技術，應用程序測試可以檢測個別關係數據庫的變化，並在源和目標之間進行比較。

關聯式資料庫變更是由測試的應用程式程式碼，使用 DML (Data Modify Language) 陳述式 (例如SQLINSERTUPDATEDELETE、或) 陳述式在來源和目標上產生，而且當應用程式使用預存程序、或在某些資料表上設定資料庫觸發程序時，或當CASCADEDELETE用來保證參照完整性時，也會間接觸發其他刪除。

資料集比較

「應用程式測試」會自動比較在來源 (記錄) 和目標重新執行) 系統上產生的參照和重新執行資料集。

若要比較資料集：

1. 從源和目標上相同的輸入數據（數據集，數據庫）開始。
2. 在來源系統 (大型主機) 上執行測試案例。
3. 擷取產生的資料集，並將其上傳到 Amazon S3 儲存貯體。您可以將輸入資料集從來源傳輸到 AWS 使用CDC分錄、畫面和資料集。
4. 指定上傳測試案例時上傳大型主機資料集的 Amazon S3 儲存貯體的位置。

重新執行完成之後，「應用程式測試」會自動比較輸出參照和目標資料集，顯示記錄是否相同、對等、不同或遺漏。例如，相對於工作負載執行時刻的日期欄位 (第一天 + 1、當月結束等) 會自動被視為相等的日期欄位。此外，您可以選擇性地定義對等規則，以便不相同的記錄仍具有相同的業務意義，並標記為對等。

比較狀態

應用程式測試使用下列比較狀態：IDENTICALEQUIVALENT、和DIFFERENT。

IDENTICAL

來源和目標資料完全相同。

EQUIVALENT

來源和目標資料包含視為對等項目的錯誤差異，例如日期或時間戳記，這些差異在相對於工作負載執行的時刻時不會影響功能對等。您可以定義對等規則來識別這些差異。當所有重播的測試套件與其參考測試套件相比顯示IDENTICAL或的狀態時EQUIVALENT，您的測試套件沒有顯示差異。

DIFFERENT

來源和目標資料包含差異，例如資料集中不同數目的記錄，或相同記錄中的不同值。

等價規則

一組規則，用於識別可視為等效結果的錯誤差異。離線功能等價測試 (OFET) 不可避免地會導致來源和目標系統之間的某些結果差異。例如，更新時間戳記因設計而異。等價規則說明如何調整這些差異，並避免在比較時誤報。例如，如果某個日期在特定資料欄中為 Runtime + 2 天，則對等規則會對其進行描述，並接受目標系統上執行時間 + 2 天的時間，而非嚴格等於參照上傳中相同欄的值。

最終狀態資料集比較

已建立或修改的資料集結束狀態，包括從其初始狀態對資料集所做的所有變更或更新。對於數據集，應用程序測試在測試用例運行結束時查看這些數據集中的記錄，並比較結果。

狀態進度資料庫比較

對數據庫記錄所做的更改作為單獨的序列DML（刪除，更新，插入）語句的比較。「應用程式測試」會比較來源資料庫與目標資料庫的個別變更 (插入、更新或刪除資料表的資料列)，並識別每個變更的差異。例如，個別INSERT陳述式可用於在資料表中插入與目標資料庫相比，來源資料庫上具有不同值的資料列。

功能等價 (FE)

如果兩個系統在所有可觀察到的操作產生相同的結果，給定相同的輸入數據，則被認為在功能上等同。例如，如果相同的輸入資料產生相同的輸出資料 (透過畫面、資料集變更或資料庫變更)，就會將兩個應用程式視為相同的功能。

在線屏幕比較

當目標系統在中的 AWS Blu Age 執行階段下執行時，將 3270 大型主機螢幕的輸出與現代化應用程式網頁螢幕的輸出進行比較。 AWS 雲端它會比較大型主機 3270 螢幕的輸出與重新裝載的應用程式的 3270 螢幕，當目標系統在 Micro Focus 執行階段中執行時。 AWS 雲端

重播資料

重新顯示資料是用來描述透過在目標測試環境中重新播放測試套件所產生的資料。例如，在 AWS 大型主機現代化服務應用程式上執行測試套件時，就會產生重新顯示資料。然後將重播資料與從來源擷取的參考資料進行比較。每次在目標環境中重新執行工作負載時，都會產生新一代的重新執行資料。

參考資料

參考資料是用來描述在來源大型主機上擷取的資料。它是到哪個重播（目標）生成的數據將被比較的引用。通常，對於大型主機上創建參考數據的每個記錄，都會有許多重播。這是因為使用者通常會擷取大型主機上應用程式的正確狀態，並在目標現代化應用程式上重新執行測試案例，以驗證對等性。如果發現錯誤，它們是固定的，並再次重播測試用例。通常，多個循環的重播，修復錯誤，並再次重播以驗證發生。這就是所謂的捕獲一次，重播多次測試範例。

上傳、重播和比較

應用程序測試分三個步驟操作：

- 上傳：擷取在大型主機上針對測試案例的每個測試案例建立的參考資料。這些可能包括 3270 個線上畫面、資料集和資料庫記錄。
  - 對於線上 3270 螢幕，您必須使用 Blu Insights 終端機模擬器來擷取來源工作負載。如需詳細資訊，請參閱 Blu 洞察文件。
  - 對於資料集，您需要使用一般工具 (例如FTP大型主機現代化的資料集傳輸服務部分)，擷取大型主機上每個測試案例所產生的 AWS 資料集。
  - 對於資料庫變更，您可以使用精確的AWS大型主機現代化資料複寫來擷取和產生包含變更的CDC日誌。
- 重新執行：測試套件會在目標環境中重新顯示。在測試套件中指定的所有測試用例都運行。由單個測試用例創建的指定數據類型，例如數據集，關係數據庫更改或 3270 屏幕，將通過自動化捕獲。這些資料稱為重播資料，會與上傳階段所擷取的參考資料進行比較。

> **Note**
>
> 關聯式資料庫變更需要初始條件 CloudFormation 範本中的DMS特定組態選項。

- 比較：對源測試參考數據進行比較，並對目標重播數據進行比較，結果將顯示為相同，不同，等效或缺少的數據。

差異

指出透過資料比較偵測到參照和重新執行資料集之間的差異。例如，線上 3270 畫面中的欄位顯示與來源大型主機與目標現代化應用程式之間的商業邏輯角度不同的值，將會被視為差異。另一個範例是在來源和目標應用程式之間不相同的資料集中進行上傳。

等值

對等記錄是參考和重新顯示資料集之間不同的記錄，但不應該被視為與商務邏輯觀點不同的記錄。例如，包含資料集產生時間戳記的記錄 (工作負載執行時間)。使用可自訂的對等項規則，您可以指示「應用程式測試」將這類誤判差視為對等項目，即使它顯示參照與重新執行資料之間的不同值也一樣。

來源應用

要與之比較的來源大型主機應用程式。

目標應用

新的或修改過的應用程式，在其上進行測試，這些應用程式會與來源應用程式進行比較，以偵測任何瑕疵，並達到來源與目標應用程式之間的功能等效性。目標應用程式通常在 AWS 雲端中執行。

AWS 大型主機現代化應用程式測試先決條件

AWS 大型主機現代化中的大型 AWS 主機現代化應用程式測試功能可讓您針對移轉專案執行自動化功能等效測試。若要準備在 AWS 大型主機現代化主控台中使用應用程式測試，請執行下列動作：

1. 定義測試用例：為目標應用程序定義要以特定順序運行和重播的測試的基本單元。如需如何建立測試案例的其他資訊，請參閱the section called “在應用測試中創建測試用例”。
2. 準備 CloudFormation 範本和輸入資料：建立用於佈建目標測試環境的 CloudFormation 範本。此範本中的變數將用於在您的 AWS 大型主機現代化應用程式中新增輸入資料和輸出變數名稱。如需詳細資訊，請參閱使用指南中的AWS CloudFormation 使用 AWS CloudFormation 範本。

3. 確保大型主機存取和資料擷取：確認您可以存取來源大型主機。這也可確保您可以擷取並上傳大型主機上執行之應用程式所產生的來源資料。

應用程式測試主控台

AWS 大型主機現代化應用程式測試主控台可協助您建立測試案例、測試套件和測試環境組態。

主題

- 在 AWS 大型主機現代化應用程式測試中建立測試案例
- 在 AWS 大型主機現代化應用程式測試中建立測試套件
- 在 AWS 大型主機現代化應用程式測試中建立測試環境組態

在 AWS 大型主機現代化應用程式測試中建立測試案例

測試案例是代表工作流程中某個動作的原子單元。如需各種概念的其他資訊，請參閱???。

> **Important**
>
> 在運行測試用例之前，您需要先創建至少一個測試環境配置。若要建立您的第一個環境組態，請參閱the section called “在應用程序測試中創建測試環境配”。

主題

- 創建 Batch 測試用例
- 創建一個在線 3270 屏幕測試用例

創建 Batch 測試用例

Batch 測試用例允許您提交批次以讀取、處理和修改或生成新的業務數據（數據庫和/或數據集記錄）。

若要建立 Batch 測試案例

1. 開啟 AWS 大型主機現代化應用程式測試主控台，網址為。https://console.aws.amazon.com/apptest/
2. 在選取 AWS 區域 器中，選擇提供應用程式測試的區域。

> **Note**
>
> 應用程式測試目前僅在美國東部 (維吉尼亞北部)、亞太區域 (雪梨)、歐洲 (法蘭克福) 和南美洲 (聖保羅) 區域提供。

3. 在左側導覽窗格中，選取 [測試案例]。
4. 在定義測試用例中，輸入您的測試用例名稱和可選描述。在測試用例類型下選擇 Batch。
5. 選擇 Next (下一步)。
6. (選擇性) 在 [指定批次JCL參數] 頁面上，新增 JCL (工作控制語言) 名稱和工作參數 (名稱和值)。
7. 選擇 Next (下一步)。
8. 在要擷取的資料來源頁面上，您可以選擇關聯式資料庫變更、資料集或兩者。
   - 當您希望測試案例修改資料庫記錄時，請選擇 [關聯式資料庫變更]。
   - 當您希望測試案例修改資料集時，請選擇 [資料集]。在「輸出資料集」下，新增輸出資料集的名稱。

   > **Note**
   >
   > 您可以新增多個資料集。
9. 選擇 Next (下一步)。
10. 在 [檢閱並建立] 頁面上，檢閱所有資訊並選擇 [建立測試案例]。

創建一個在線 3270 屏幕測試用例

在線 3270 屏幕測試用例允許您運行交互式屏幕對話框（3270）以讀取，修改或生成新的業務數據（數據庫和/或數據集記錄）。

建立線上 3270 螢幕測試案例

1. 開啟 AWS 大型主機現代化應用程式測試主控台，網址為。https://console.aws.amazon.com/apptest/
2. 在選取 AWS 區域 器中，選擇提供應用程式測試的區域。

> **Note**
>
> 應用程式測試目前僅在美國東部 (維吉尼亞北部)、亞太區域 (雪梨)、歐洲 (法蘭克福) 和南美洲 (聖保羅) 區域提供。

3. 在左側導覽窗格中，選取 [測試案例]。
4. 在定義測試用例中，輸入您的測試用例名稱和可選描述。選擇在線 3270 屏幕下測試用例類型。
5. 選擇 Next (下一步)。

> **Note**
>
> 在線 3270 屏幕不需要您指定JCL參數。

6. 選擇 Next (下一步)。
7. 在 [要擷取的資料來源] 頁面上，預設選取項目為 [線上 3270] 畫面。此外，您可以選擇 [關聯式資料庫變更] 和 [資料集]。

   - 當您希望測試案例修改資料庫記錄時，請選擇 [關聯式資料庫變更]。
   - 當您希望測試案例修改資料集時，請選擇 [資料集]。在「輸出資料集」下，新增輸出資料集的名稱。

   > **Note**
   >
   > 您可以新增多個資料集。

8. 選擇 Next (下一步)。
9. 在 [檢閱並建立] 頁面上，檢閱所有資訊並選擇 [建立測試案例]。

在 AWS 大型主機現代化應用程式測試中建立測試套件

測試套件是一系列按順序運行的測試用例。測試套件對於重播測試用例很重要。

> **Important**
>
> 在創建測試套件之前，您需要至少有一個測試用例。您可以使用、建立您的第一個測試案例the section called “在應用測試中創建測試用例”。

如需各種概念的其他資訊，請參閱the section called “應用程式測試概”。

主題

- 建立測試套件
- 上傳參考資料
- 重播與比較

建立測試套件

測試套件允許您運行不同的測試用例，並在以後重播並進行比較。

若要建立測試套件

1. 開啟 AWS 大型主機現代化應用程式測試主控台，網址為。https://console.aws.amazon.com/apptest/
2. 在選取 AWS 區域 器中，選擇提供應用程式測試的區域。

   > **Note**
   >
   > 應用程式測試目前僅在美國東部 (維吉尼亞北部)、亞太區域 (雪梨)、歐洲 (法蘭克福) 和南美洲 (聖保羅) 區域提供。

3. 在左側導覽窗格中，選取 [測試案例]。
4. 選擇 [建立測試套件]。
5. 在「創建測試套件」部分中，從測試用例庫中查找測試用例，然後選擇添加選定的測試用例。

   > **Note**
   >
   > 您最多可以在一個測試套件中添加 20 個測試用例。

6. 在「測試套件」窗格中，輸入您的測試套件名稱和可選說明。此外，從託管運行時或非託管運行時中進行選擇，這將定義測試套件如何配置和取消配置大型主機現代化應用程序。 AWS 選擇性地新增 AWS 大型主機現代化匯入資料集 S3。JSON URI
7. 在「添加的測試用例」部分中，按照要上傳的順序堆疊測試用例並重播它們。
8. 選擇 Create test suite (建立測試套件)。

上傳參考資料

將大型主機參考資料上傳至 AWS 應用程式測試。您只需要第一次保存上傳的參考數據。測試服務可以重複使用來源上傳的結果，並與目標上的重播結果進行連續比較。

若要上傳參考資料

1. 在「測試套件」區段中，選擇要上傳參考資料的測試套件。
2. 選擇上傳。
3. 在 [上傳參考資料] 頁面上，選取您要重播的測試案例。完成資料擷取日期、資料庫變更日誌 S3 位置、資料集 S3 位置和選擇上傳的欄位。

重播與比較

重新執行和比較程序會將您的測試案例與目標測試環境產生關聯，並執行應用程式。在執行重播程序之前，您必須先上傳資料。

若要重新執行與比較

1. 從「測試套件」區段中，選擇要重播的測試套件。
2. 選擇重播並進行比較。
3. 在「重新執行和比較總覽」頁面上，選取您的測試環境組態並複查資訊。編輯功能允許您編輯任何測試環境配置字段。您也可以找到 AWS CloudFormation 參數。
4. 在「要重播的測試用例」部分下，選擇測試用例並按照要重播的順序放置它們。
5. 選擇重播並進行比較。

在 AWS 大型主機現代化應用程式測試中建立測試環境組態

測試環境配置允許您設置初始一組數據和配置參數（或資源） AWS CloudFormation ，以使測試運行可重複。

如需各種概念的其他資訊，請參閱the section called “應用程式測試概”。

建立測試環境設定

設定測試環境以重新執行並比較「應用程式測試」中的測試案例。

設定測試環境組態

1. 開啟 AWS 大型主機現代化應用程式測試主控台，網址為。https://console.aws.amazon.com/apptest/
2. 在選取 AWS 區域 器中，選擇提供應用程式測試的區域。

> **Note**
>
> 應用程式測試目前僅在美國東部 (維吉尼亞北部)、亞太區域 (雪梨)、歐洲 (法蘭克福) 和南美洲 (聖保羅) 區域提供。

3. 在左側導覽窗格中，選取 [測試環境組態]。
4. 選擇 [建立測試環境組態]。
5. 在 [建立測試環境設定] 窗格中，輸入名稱和說明。同時新增包含應用程式測試 CloudFormation 範本的 Amazon S3 儲存貯體。此外，您可以新增將在 CloudFormation 堆疊建立期間使用的 CloudFormation 輸入參數。
6. 指定將受此測試組態影響的AWS大型主機現代化應用程式。新增AWS大型主機現代化應用程式 ID、執行階段引擎的輸出變數名稱 (AWSBlu Age 非受管或 Micro Focus 管理)。

> **Note**
>
> AWS大型主機現代化應用程式 ID 的輸出變數名稱應與用於堆疊建立之 CloudFormation 範本的輸出變數名稱相符。

> **Important**
>
> AWSBlu Age 非受管理執行階段也會要求您指定VPC端點服務 ID 的輸出變數名稱、監聽程式連接埠的輸出變數名稱，以及 name 的輸出變數名 WebApp 稱。這些名稱應與 CloudFormation 範本中的輸出變數名稱相符。

7. (選擇性) 可以為 Database Migration Service () 任務 Amazon 資源名稱 (DMS) 定義其他屬性，例如輸出變數名稱，該工作用於擷取關聯式資料庫變更。ARN另一個屬性是來源資料庫 DDL S3 URI。

> **Important**
>
> 輸出變數名稱應與 CloudFormation 範本中的變數名稱相符。

8. (選擇性) 自訂金鑰管理服務 (KMS) 金鑰。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的管理客戶受管金鑰的存取。
9. 選擇 [建立測試環境組態]。

教學課程：在 AWS 大型主機現代化應用程式測試中設定 CardDemo 範例應用程式

在本教學課程中，您會建立一個 AWS CloudFormation 堆疊，協助您使用 Micro Focus on 大型主機現代化受管理服務，以及包括大型 AWS 主機現代化應用程式測試等 AWS 功能來設定重新平台的 CardDemo 範例應用程式。本教學課程說明可用來建立堆疊的 AWS CloudFormation 範例範本。我們還提供必要的應用程序工件的壓縮文件。範例範本會佈建資料庫、執行階段環境、應用程式和完全隔離的網路環境。

此範本會建立數個 AWS 資源。如果您從此範本建立堆疊，將會向您收取費用。

必要條件

- 下載並解壓縮`IC3-card-demo-zip`和`datasets_Mainframe_ebcdic.zip`。這些檔案包含與應用 AWS 程式測試搭配使用的範例和範例資料集。 CardDemo
- 建立 Amazon S3 儲存貯體來存放 CardDemo 檔案和其他成品。例如：`my-carddemo-bucket`。

步驟 1：準備設定 CardDemo

上傳 CardDemo 範例檔案並編輯將建立 CardDemo應用程式的 AWS CloudFormation 範本。

1. 將先前解壓縮的`IC3-card-demo`資料夾`datasets_Mainframe_ebcdic`和資料夾上傳至值區。
2. 從值區下載`aws-m2-math-mf-carddemo.yaml` AWS CloudFormation 範本。它位於`IC3-card-demo`文件夾中。
3. 編輯`aws-m2-math-mf-carddemo.yaml` AWS CloudFormation 範本，如下所示：
   - 將`BucketName`參數變更為您先前定義的值區名稱，例如`my-carddemo-bucket`。

- `ImportJsonPath`將變更為檔案儲存貯體中的位`mf-carddemo-datasets-import.json`置。例如，`s3://my-carddemo-bucket/IC3-card-demo/mf-carddemo-datasets-import.json`更新此值可確保輸出`M2ImportJson`具有正確的值。
- (選擇性) 調整`EngineVersion`和`InstanceType`參數以符合您的標準。

> **Note**
>
> 請勿修改`M2EnvironmentId`和`M2ApplicationId`輸出。應用程序測試使用這些值來定位與它將交互的資源。

步驟 2：建立所有必要的資源

執行您的自訂 AWS CloudFormation 範本以建立成功完成此教學課程所需的所有資源。該模板設置應用程 CardDemo 序，以便您可以在測試中使用它。

1. 登入 AWS CloudFormation 主控台並選擇 [建立堆疊]，然後選擇 [使用新資源 (標準)]。
2. 在先決條件-準備範本中，選擇範本已準備就緒。
3. 在「指定範本」中，選擇「上傳範本檔案」，然後選擇「選擇檔案」。
4. 導覽至您下載的位置`aws-m2-math-mf-carddemo.yaml`並選擇該檔案，然後選擇「下一步」。
5. 在指定堆棧詳細信息中為堆棧提供名稱，以便您可以輕鬆地在列表中找到它，然後選擇下一步。
6. 在 [設定堆疊選項] 中，保留預設值並選擇 [下一步]。
7. 在「檢閱」中，勾選 AWS CloudFormation 正在為您建立的項目，然後選擇「提交」。

創建堆棧大約需要 10-15 分鐘。 AWS CloudFormation

> **Note**
>
> 範本設定為在其建立的資源名稱中附加唯一的尾碼。這表示您可以同時建立此堆疊範本的多個執 parallel 個體，這是應用程式測試的關鍵功能，可讓您同時執行多個測試套件。

步驟 3：部署並啟動應用程式

部署為您 AWS CloudFormation 建立的 CardDemo 應用程式，並確定應用程式正在執行。

1. 開啟「 AWS 大型主機現代化」主控台，然後從左側導覽列選擇「應用程式」。
2. 選擇 CardDemo 應用程序，它被命名為類似的東西`aws-m2-math-mf-carddemo-abc1d2e3`。
3. 選擇動作，然後選擇部署應用程式。
4. 在環境中，選擇與應用程式對應的執行階段環境。它將在名稱的末尾附加相同的唯一標識符。例如：`aws-m2-math-mf-carddemo-abc1d2e3`。
5. 選擇部署。等到應用程式部署成功，並處於 [就緒] 狀態。
6. 選擇應用程式，然後選擇動作和啟動應用程式。等待應用程式處於 [執行中] 狀態。
7. 在應用程式詳細資訊頁面中，複製連接埠和DNS主機名稱 (Port) 和主機名稱，以連線到執行中的應用程式。

步驟 4：匯入初始資料

若要使用 CardDemo 範例應用程式，您必須匯入一組初始的資料。完成下列步驟。

1. 下載 `mf-carddemo-datasets-import.json` 檔案。
2. 在您偏好的文字編輯器中編輯檔案。
3. 找到`s3Location`參數並更新值，以指向您建立的 Amazon S3 儲存貯體。
4. 對所有出現的項目進行相同的變更`s3Location`，然後儲存檔案。
5. 登入 Amazon S3 主控台並導覽至您之前建立的儲存貯體。
6. 上傳自訂`mf-carddemo-datasets-import.json`檔案。
7. 開啟「 AWS 大型主機現代化」主控台，然後從左側導覽列選擇「應用程式」。
8. 選擇 CardDemo 應用程式。
9. 選擇 [資料集]，然後選擇 [匯入]。
10. 導覽至 Amazon S3 中您上傳自訂JSON檔案的位置，然後選擇「提交」。

此工作匯入 23 個資料集。若要監視匯入工作的結果，請檢查主控台。成功匯入所有資料集後，請連線至應用程式。

> **Note**
>
> 當您在應用程式測試中使用此範本時，輸出`M2ImportJson`會自動處理匯入程序。

步驟 5：Connect 到 CardDemo應用程序

使用您選擇的 3270 模擬器 Connect 到 CardDemo 示例應用程序。

- 執行應用程式時，請使用 3270 模擬器連線至應用程式，並視需要指定DNS主機名稱和連接埠名稱。

  例如，如果您使用的是開源 c3270 模擬器，則您的命令看起來像這樣：

  ```
  c3270 -port port-number DNS-hostname
  ```

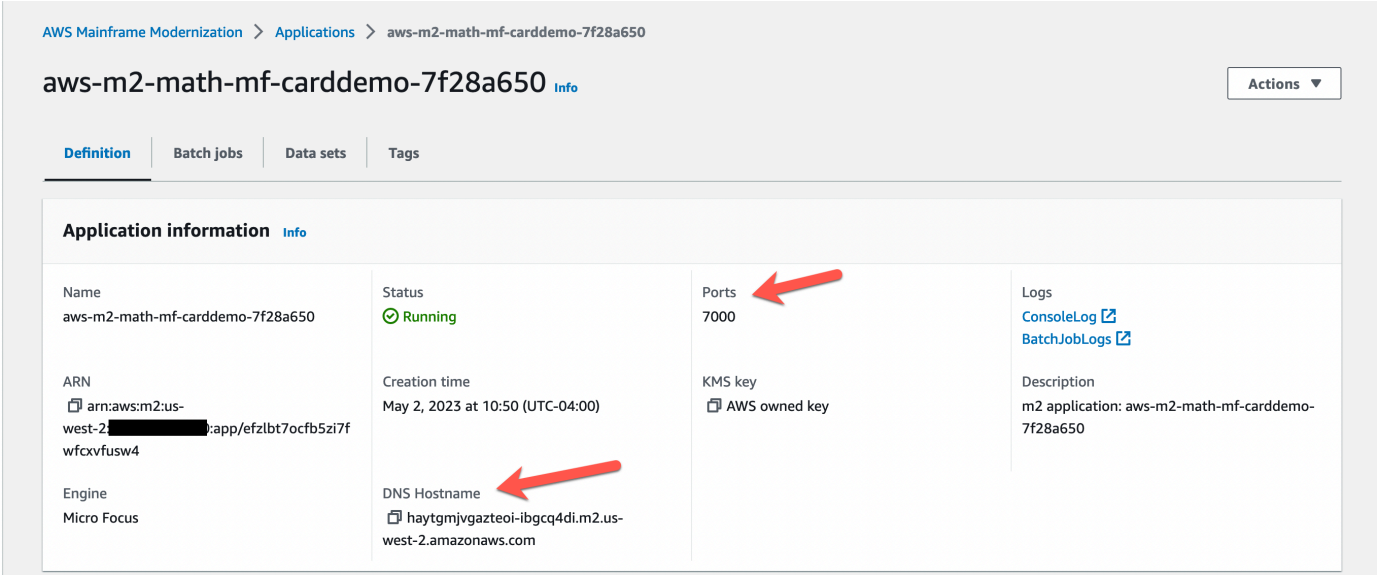
  port

  在應用程式詳細資訊頁面上指定的連接埠。例如：6000。

  Hostname (主機名稱)

  應用程式詳細資訊頁面上指定的DNS主機名稱。

  下圖顯示了在何處可以找到端口和DSN主機名。



教學課程：使用 Amazon 上部署的 AWS Blu AWS Age，在大型主機現代化應用 CardDemo 程式測試中重播和比較 EC2

在本教學中，您將完成必要步驟，以重播測試工作負載並比較在 Amazon 上部署的 AWS Blu Age 上執行的 CardDemo 應用程式EC2。

步驟 1：獲取 AWS 藍光時代 Amazon EC2 Amazon 機器圖像（AMI）

請按照 AWS Blu Age 運行時（在 Amazon 上EC2）安裝教程中的說明進行操作，了解在 Amazon EC2 AMI 上訪問 AWS Blu Age 所需的入門步驟。

步驟 2：使用 AWS 藍光時代啟動 Amazon EC2 實例 AMI

1. 設定您的 AWS 認證。
2. 從 Amazon Amazon S3 識別 3.5.0 亞馬遜EC2AMI二進位檔案 (CLI僅限/AWS 藍光時代版本) 的位置：

```
aws s3 ls s3://aws-bluage-runtime-artifacts-xxxxxxx-eu-west-1/
aws s3 ls s3://aws-bluage-runtime-artifacts-xxxxxxx-eu-west-1/3.5.0/AMI/
```

> **Note**
>
> 應用程序測試功能僅適用於產品中的 4 個地區（us-east-1，SA-東 1，eu-central-1 和 ap-southeast-2）。

3. 使用以下命令恢復您的帳戶AMI中：

```
aws ec2 create-restore-image-task --object-key 3.5.0/AMI/ami-0182ffe3b9d63925b.bin
 --bucket aws-bluage-runtime-artifacts-xxxxxxx-eu-west-1 --region eu-west-1 --name
 "AWS BLUAGE RUNTIME AMI"
```

> **Note**
>
> 替換 AMI bin 文件名和要創建的區域AMI。

4. 建立 Amazon EC2 執行個體後，您可以在 Amazon EC2 映像目錄中找到AMI從 Amazon S3 儲存貯體還原的正確 AMI ID。

> **Note**
>
> 在本自學課程中，AMI識別碼為 ami-0d0fafcc636fd1e6d，您必須將不同組態檔案中的這個識別碼變更為提供給您的識別碼。

1. 如果 aws ec2 create-restore-image-task 失敗，請檢查您的 Python 版本並CLI使用以下命令：

```
aws --version
```

> **Note**
>
> Python 版本必須 > = 3，並且CLI版本必須 > = 2。

2. 如果這些版本已過時，則CLI必須更新。若要更新CLI：
   a. 請遵循安裝或更新最新版本的指示AWSCLI。
   b. 使用以下命令刪除 CLI v1：

```
sudo yum remove awscli
```

   c. 並使用以下命令安裝 CLI v2：

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
 "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

   d. 最後，檢查 Python 的版本，並CLI使用以下命令：

```
aws --version
```

3. 然後，您可以重做 aws ec2 create-restore-image-task。

步驟 3：將 CardDemo 相依檔案上傳至 S3

複製資料夾資料庫、檔案系統和使用者資料的內容。下載並解壓縮 CardDemo 應用程序。在本文件中，這三個資料夾必須複製到其中一個稱為 Your-s3-bucket 的值區中。

步驟 4：載入資料庫並初始化 CardDemo應用程式

建立一個臨時 Amazon EC2 執行個體，您將用作運算資源，以產生 CardDemo應用程式所需的資料庫快照。此EC2執行個體不會執行 CardDemo 應用程式本身，而是會產生稍後使用的資料庫快照集。

首先編輯提供的 CloudFormation 模板名為 'load-and-create-ba-Snapshots.yml「。這是用來建 CloudFormation 立用來產生資料庫快照之 Amazon EC2 執行個體的範本。

1. 產生並提供將用於EC2執行個體的 EC2 key pair。如需詳細資訊，請參閱建立金鑰配對。

   範例：

   ```
   Ec2KeyPair:
       Description: 'ec2 key pair'
       Default: 'm2-tests-us-west-2'
       Type: String
   ```

2. 指定您在上一個步驟中放置資料庫資料夾的資料夾的 Amazon S3 路徑：

   ```
   S3DBScriptsPath:
       Description: 'S3 DB scripts folder path'
       Type: String
       Default: 's3://your-s3-bucket/databases'
   ```

3. 指定您在上一個步驟中放置檔案系統資料夾的資料夾的 Amazon S3 路徑：

   ```
   S3ApplicationFilesPath:
       Description: 'S3 application files folder path'
       Type: String
       Default: 's3://your-s3-bucket/file-system'
   ```

4. 指定您在上一步中放置使用者資料資料夾的資料夾的 Amazon S3 路徑：

   ```
   S3UserDataPath:
       Description: 'S3 userdata folder path'
       Type: String
       Default: 's3://your-s3-bucket/userdata'
   ```

5. 同時指定 Amazon S3 路徑，您將在其中儲存要在下一個步驟中使用的結果檔案。

   ```
   S3SaveProducedFilesPath:
       Description: 'S3 path folder to save produced files'
   ```

```
    Type: String
    Default: 's3://your-s3-bucket/post-produced-files'
```

6. 使用下列範本，使用本教學課程稍早取得的正確 AMI ID 來變更 ID：

```
BaaAmiId:
  Description: 'ami id (AL2) for ba anywhere'
  Default: 'ami-0bd41245734fd20d9'
  Type: String
```

   - 您可以選擇性地變更執行載入資料庫所建立之三個快照集的名稱 withCloudFormation。這些將在 CloudFormation 堆棧中可見，因為它正在創建，並將在本教程後面使用。請記住要注意用於資料庫快照集的名稱。

```
SnapshotPrimary:
    Description: 'Snapshot Name DB BA Primary'
    Type: String
    Default: 'snapshot-primary'

  SnapshotBluesam:
    Description: 'Snapshot Name DB BA Bluesam'
    Type: String
    Default: 'snapshot-bluesam'

  SnapshotJics:
    Description: 'Snapshot Name DB BA Jics'
    Type: String
    Default: 'snapshot-jics'
```

> **Note**
>
> 在本文件中，我們假設快照的名稱保持一致。

7. 使用「 CloudFormation 建立堆疊」按鈕和精靈執行 and CLI 或 AWS 主控台。在該過程結束時，您應該會在RDS控制台中看到三個快照，其中包含您選擇的名稱後跟一個唯一的 ID。在下一步中，您將需要這些名稱。

> **Note**
>
> RDS會在 AWS CloudFormation 範本中定義的快照名稱中加入字尾。請務必先取得完整快照名稱，然RDS後再繼續進行下一個步驟。

示例CLI命令-

```
aws cloudformation create-stack --stack-name load-and-create-ba-snapshots --
template-url https://your-apptest-bucket.s3.us-west-2.amazonaws.com/load-and-
create-ba-snapshots.yml --capabilities CAPABILITY_NAMED_IAM
```

您也可以簽入您為 S3 提供的 Amazon S3 路徑，以SaveProducedFilesPath確定資料集已正確建立。

步驟 5：啟動 AWS 藍光時代運行時 CloudFormation

用 CloudFormation 於使用 CardDemo AWS 藍光時代應用程序運行 Amazon EC2 實例。您必須`m2-with-ba-using-snapshots-https-authentication.yml`透過編輯YAML檔案或在啟動期間修改主控台中的值來取代 CloudFormation 命名中的某些變數CFN。

1. 使用下列命令修改以指定哪個帳戶將到達用於存取 AWS Blu Age 執行階段的VPC端點：`AllowedVpcEndpointPrincipals`

```
AllowedVpcEndpointPrincipals:
    Description: 'comma-separated list of IAM users, IAM roles, or AWS accounts'
    Default: 'apptest.amazonaws.com'
    Type: String
```

2. 將變數的值 SnapshotPrimaryDb SnapshotBlusamDb、和變更 SnapshotJicsDb 為快照的名稱。同時從上一個步驟中建立快照名稱RDS之後取得快照名稱。

```
SnapshotPrimary:
    Description: 'Snapshot DB cluster for DB Primary'
    Type: String
    Default: 'snapshot-primary87d067b0'

  SnapshotBluesam:
```

```
    Description: 'Snapshot DB cluster for DB Bluesam'
    Type: String
    Default: 'snapshot-bluesam87d067b0'

  SnapshotJics:
    Description: 'Snapshot DB cluster for DB Jics'
    Type: String
    Default: 'snapshot-jics87d067b0'
```

> **Note**
>
> RDS會將自己的後綴添加到快照名稱中。

3. 使用以下命令為EC2執行個體提供 Amazon EC2 key pair：

```
Ec2KeyPair:
    Description: 'ec2 key pair'
    Default: 'm2-tests-us-west-2'
    Type: String
```

4. 使用下列方法為變數BaaAmiId提供您在AMI註冊程序期間取得的 AMI ID：

```
BaaAmiId:
    Description: 'ami id (AL2) for ba anywhere'
    Default: 'ami-0d0fafcc636fd1e6d'
    Type: String
```

5. 使用下列命令提供您在上一個步驟中用來儲存產生檔案的 Amazon S3 資料夾路徑：

```
S3ApplicationFilesPath:
    Description: 'bucket name'
    Type: String
    Default: 's3://your-s3-bucket/post-produced-files'
```

6. 最後，提供 s3- 的文件夾路徑userdata-folder-path：

```
S3UserDataPath:
    Description: 'S3 userdata folder path'
    Type: String
    Default: 's3://your-s3-bucket/userdata'
```

- (選擇性) 您可以啟用 tomcat 的HTTPS模式和基本HTTP驗證。雖然默認設置也可以工作。

  > **Note**
  >
  > 默認情況下，該HTTPS模式被禁用並在參數HTTP中設置為模式 BacHttpsMode：

  例如：

  ```
  BacHttpsMode:
      Description: 'http or https for Blue Age Runtime connection mode '
      Default: 'http'
      Type: String
      AllowedValues: [http, https]
  ```

- (選擇性) 若要啟用HTTPS模式，您必須將值變更為HTTPS並提供ACM憑證，ARN方法是變更變數的值 ACMCertArn：

  ```
  ACMCertArn:
      Type: String
      Description: 'ACM certificate ARN'
      Default: 'your arn certificate'
  ```

- (選擇性) 基本驗證依預設為停用，且參數WithBacBasicAuthentication設定為 false。您可以通過將值設置為 true 來啟用它。

  ```
  WithBacBasicAuthentication:
      Description: 'false or true for Blue Age Runtime Basic Authentication '
      Default: false
      Type: String
      AllowedValues: [true, false]
  ```

7. 完成設定後，您可以使用編輯過的 CloudFormation 範本建立堆疊。

步驟 6：測試 AWS 藍光時代 Amazon EC2 實例

手動執行 CloudFormation 範本，為 CardDemo 應用程式建立 AWS Blu Age Amazon EC2 執行個體，以確保其啟動時沒有錯誤。這樣做是為了在使用 CloudFormation 範本與應用程式測試功能之前，先確

認 CloudFormation 範本和所有必要條件是否有效。然後，您可以使用應用程式測試在重播和比較期間自動建立目標 AWS Blu Age Amazon EC2 執行個體。

1. 執行 CloudFormation 建立堆疊命令來建立 AWS Blu Age Amazon EC2 執行個體，並提供您在上一步中編輯的 m2-with-ba-using-snapshots-https-驗證 .yml CloudFormation 範本：

```
aws cloudformation create-stack --stack-name load-and-create-ba-snapshots —-
template-url https://apptest-ba-demo.s3.us-west-2.amazonaws.com/m2-with-ba-using-
snapshots-https-authentication.yml --capabilities CAPABILITY_NAMED_IAM --region us-
west-2
```

> **Note**
>
> 請記住指定恢復 AWS 藍光AMI時代的正確區域。

2. 在主控台中尋找執行中的 Amazon EC2 執行個體，以確保一切正常運作。使用工作階段管理員 Connect 線到它。
3. 連線到 Amazon EC2 執行個體後，請使用下列命令：

```
sudo su
cd /m2-anywhere/tomcat.gapwalk/velocity/logs
cat catalina.log
```

4. 請確定記錄中沒有例外或錯誤。
5. 接下來，使用以下命令檢查應用程式是否正在回應：

```
curl http://localhost:8080/gapwalk-application/
```

您會看到訊息，「Jics 應用程式正在執行中。」

步驟 7：驗證之前的步驟是否正確完成

在接下來的幾個步驟中， AWS 我們將使用大型主機現代化應用程式測試來重新顯示和比較應用程式所建立的資料集。 CardDemo 這些步驟需要成功完成本教學課程中的所有先前步驟。請先驗證下列項目，再繼續：

1. 您已經通過 AWS CloudFormation 模板在 Amazon EC2 實例上成功創建了 AWS 藍光時代。
2. Amazon 上 AWS 藍光時代的 Tomcat 服務EC2已啟動並運行，沒有例外。

當執行個體與應用程式一起EC2執行時， CardDemo 請在「應用程式測試」主控台上完成下列步驟，以執行重新執行和比較批次資料集。

步驟 8：創建測試用例

在此步驟中，您會建立將用來比較在卡片示範應用程式中建立的資料集的測試案例。

1. 創建一個新的測試用例。給它一個名稱和描述。
2. 指定CREASTMT.JCL為名JCL稱。
3. 將以下數據集添加到測試用例定義中：

名稱	CCSID	RecordFormat	RecordLength
AWS. 平方米 CARDDEMO。 STATEMNT.PS	「037」	FB	80
AWS. 平方米 CARDDEMO。 STATEMNT。 HTML	「037」	FB	100

> **Note**
>
> 您的JCL名稱和資料集詳細資料必須相符。

步驟 9：建立測試套件

1. 創建一個新的測試套件，並為其提供名稱和說明。
2. 將您在上一步中創建的測試用例添加到測試套件中。
3. 建立測試套件後，在大型主機上擷取測試案例，並將大型主機參考資料上傳至應用程式測試。 AWS
4. 選擇 Create test suite (建立測試套件)。

步驟 10：建立測試環境設定

1. 創建一個新的測試環境配置，並為其提供名稱和描述。
2. 新增您的 CloudFormation 範本。您還可以從 CloudFormation 模板中添加輸入參數名稱和值。
3. 選擇 AWS 大型主機現代化服務 AWS Blu Age 非託管作為您的運行時間。
4. 新增 AWS 大型主機現代化應用程式 ID 名稱的輸出變數名稱、VPC端點服務 ID 的輸出變數名稱、接聽程式連接埠的輸出變數名稱，以及 name 的輸出變數名稱。 WebApp

   > **Note**
   >
   > 這些欄位的名稱應與 CloudFormation 範本的輸出變數名稱相符，這些變數名稱將在堆疊建立期間從 AWS 大型主機現代化傳回。

5. (選擇性) 為 (資料庫移轉服務) 工作ARN和來源資料庫 DMS DDL (資料庫定義語言) S3 URI 位置選擇輸出變數名稱。
6. (選擇性) 自訂金鑰管理服務 (KMS) 金鑰。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的管理客戶受管金鑰的存取。
7. 選擇 [建立測試環境組態]。

步驟 11：在測試套件中上傳輸入數據

在此步驟中，您會在來源上執行測試案例。要做到這一點：

1. 下載並執行源自應用程式大型主機執行的資料集。 CardDemo
2. 將解壓縮的資料夾上傳到您的 Amazon S3 儲存貯體。此 Amazon S3 儲存貯體必須與其他應用程式測試資源位於相同的區域。

   > **Note**
   >
   > 應該有兩個文件，其名稱與先前測試用例中傳遞的數據集名稱匹配。

3. 在 [測試套件總覽] 頁面上，選擇 [上傳] 按鈕。
4. 在 [上傳參考資料] 頁面上，指定您將從來源大型主機取得之資料集上傳到的 Amazon S3 位置。
5. 選擇「上傳」以開始上傳程序。

> **Note**
>
> 在執行重播和比較之前，請等待錄製完成。

步驟 12：重播和比較

在 Amazon EC2 環境上的目標 AWS AWS 藍光時代中運行測試套件和測試用例。應用程式測試會擷取重新執行產生的資料集，並將它們與大型主機上記錄的參考資料集進行比較。

1. 選擇重播並進行比較。它應該需要大約三分鐘來創建 CloudFormation 堆棧，並執行比較。

一旦一切都完成，你應該有比較結果與為此演示的目的故意創建了一些差異。

AWS 大型主機現代化應用程式測試支援的資料集程式碼頁

使用下表判斷「應用 AWS 程式測試」是否支援資料的編碼字元集識別碼 (CCSID)。如果您的資料使用不受支援的資料CCSID，我們建議您將其轉換為支援資料，CCSID或聯絡我們尋求協助。

CCSID	字元集	描述
37	IBM037, IBM -037,	主辦機構：USA、加拿大（ESA）、荷蘭、葡萄牙、巴西、澳洲、紐西蘭
273	IBM273, IBM -273	主辦機構：奧地利、德國
277	IBM277, IBM -277	主辦機構：丹麥、挪威
278	IBM278, IBM -278	主辦機構：芬蘭、瑞典
280	IBM280、IBM -280	主機：意大利
284	IBM284, IBM -284	主辦單位：西班牙、拉丁美洲(西班牙文)
285	IBM285, IBM -285	主辦機構：英國

CCSID	字元集	描述
297	IBM297, IBM -297	主辦機構：法國
300	IBM-300	JAPAN資料庫 EBCDIC
301	IBM-301	電腦資料:日本 DB
437	IBM437, IBM -437, 美國-ASCII,, 437ASCII, 美國 ASCII	PC 數據：PC 基地USA，許多其他國家
500	IBM500, IBM -500, 百分比	主辦機構：比利時、加拿大 (AS/400)、瑞士、國際拉丁 1
720	IBM-720	MSDOS ARABIC
737	IBM-737, 十分之一 IBM737	MSDOS GREEK
775	IBM775, IBM -775	MSDOS BALTIC
808	IBM-808	PC 數據：西里爾文，俄羅斯，歐元
813	ISO-8859-7, _7 ISO8859	ISO希臘
819	ISO-8859-1, _1 ISO8859	ISO8859-1：拉丁裔 1 國家
833	IBM-833	KOREAN EBCDIC
834	IBM-834, X-IBM834	KOREAN資料庫 EBCDIC
835	IBM-835	T-CHINESE 分貝 EBCD
836	IBM-836	S-CHINESE EBCDIC
837	IBM-837	S-CHINESE EBCDIC
850	IBM850, IBM -850	PC 數據：拉丁 -1 國家
855	IBM855, IBM -855	PC 資料:斯拉夫文
856	IBM-856, X-IBM856,	PC 資料:希伯來文

CCSID	字元集	描述
858	IBM858, IBM -858,	PC 數據：拉丁 1 國家，歐元
859	IBM-859	電腦資料：LATIN-9
860	IBM860, IBM -860	PC 資料:葡萄牙文
861	IBM861, IBM -861	PC 資料：冰島
862	IBM862, IBM -862,	PC 數據：希伯來語（遷移）
863	IBM863, IBM -863	PC 資料：加拿大
865	IBM865, IBM -865	PC 數據：丹麥/挪威
866	IBM866, IBM -866	PC 資料：西里爾文、俄羅斯
867	IBM-867	PC 數據：希伯來語與歐元
870	IBM870, IBM -870,	主持人：拉丁裔 2 多種語言
871	IBM871, IBM -871	主辦機構：冰島
874	x-IBM874	PC 資料：泰文
875	IBM-875, X-IBM875,	主辦機構：希臘
897	IBM-897	個人電腦資料：日本 SB
912	ISO-8859-2, _2 ISO8859	ISO拉丁語 2-多種語言
915	ISO-8859-5, _5 ISO8859	ISO8859-5: 斯拉夫文 (斯拉夫文)
916	ISO-8859-8, _8 ISO8859	ISO希伯來文
918	IBM918, IBM -918	主機：烏爾都語
920	ISO-8859-9, _9 ISO8859	ISO拉丁五號 (-128, 土耳其 TS-5881) ECMA

CCSID	字元集	描述
921	IBM-921, X-IBM921,	PC 數據：拉脫維亞，立陶宛
922	IBM-922, X-IBM922,	PC 數據：愛沙尼亞
923	ISO-8859-15, 九十三, _15 ISO8859 FDIS	ISO拉丁裔
924	IBM-924	ISO拉丁裔
927	IBM-927	電腦資料：T-中文
930	IBM-930, X-IBM93 0,	片假名主持人：擴展。SBCS 漢字主機：DBCS包括 4370 個使用者定義的字元
932	IBM-932	PC 數據：日本混合
933	IBM-933, X-IBM933,	主機：已延伸SBCS。主機：DBCS包括 1880 個使用者定義的字元和 11172 個完整的韓文字元
935	IBM-935, X-IBM935,	主機：已延伸SBCS。主機：DBCS包括 1880 個使用者定義的字元。
937	IBM-937, X-IBM937,	主機：已延伸SBCS。主機：DBCS包括 6204 個使用者定義字元
939	IBM-939, X-IBM939,	拉丁主機：擴展SBCS。漢字主機：DBCS包括 4370 個使用者定義的字元。
942	IBM-942, IBM -942 度, X-, X-, 共 942IBM942, IBM942C	個人電腦資料：已延伸 SBCS。電腦資料：DBCS包括 1880 個使用者定義字元

CCSID	字元集	描述
943	IBM-943, IBM -943, 轉移 _, 視窗 -31 日JIS, 視窗 932, X-, X-, IBM943 IBM943C MS932	個人電腦數據：SBCS. 電腦資料：DBCS適用於開放環境，包括 1880使用者定IBM義字元
947	IBM-947	T-CHINESE BIG 五
948	IBM-948, X-IBM948,	個人電腦資料：已延伸 SBCS。電腦資料：DBCS包括 6204 個使用者定義字元
949	IBM-949, IBM -949 度, X-, X-IBM949, IBM949C	IBMKS 代碼-電腦數據：S BCS. IBMKS 代碼-PC 數據：DBCS包括 1880 個用戶定義的字符
950	五大牌, IBM -950, X-IBM95 0,	電腦資料：SBCS(IBMBIG5)。電腦資料：DBCS包括已選取 566 個CNS、6204 個使用者 IBM定義的字元
951	IBM-951	個人電腦資料:IBMKS
954	EUC-太平IBM紳士 IBM	一級JIS賽：羅馬。一級：JISX208-1990。一級：片JIS 假名。國一 JIS
964	EUC-台灣, IBM -964, X-, IBM964	G0:ASCII. CNS一級飛機 一級：一CNS六四三平面
970	EUC-韓國, X-IBM97 0,	G0:ASCII. G1：KSCX56 01-1989 包含 1880 個使用者定義的字元
971	IBM-971	KOREAN EUC
1006	IBM-1006, x-IBM1 006,	ISO-8: 烏爾都語

CCSID	字元集	描述
1025	IBM-1025, X-IBM1 025,	主持人：西里爾多語言
1026	IBM1026, IBM -1026,	主持人:拉丁 -5 (土耳其)
1027	IBM-1027	JAPAN LATIN EBCD
1041	IBM-1041	PC 資料：日本
1043	IBM-1043	電腦資料：T-中文
1046	IBM-1046, IBM -1046 秒, IBM1	ARABIC-個人電腦
1047	IBM1047, IBM -1047	主持人：拉丁 -1
1051	惠普羅曼 8	惠普 EMULATION
1088	IBM-1088	個人電腦資料：韓國 KS
1089	ISO-8859-6, _6 ISO8859	ISO阿拉伯語
1097	IBM-1097, X-IBM1 097,	主持人：波斯語
1098	IBM-1098, 十九, 十IBM1九	PC 數據：波斯語
1112	IBM-1112, X-IBM1112,	主機：拉脫維亞，立陶宛
1114	IBM-1114	電腦資料：T-CH SB
1115	IBM-1115	個人電腦資料：S-CH GB
1122	IBM-1122, 十二, 十二 IBM1122	主機：愛沙尼亞
1123	IBM-1123, X-IBM1123,	主持人：西里爾烏克蘭
1124	IBM-1124, X-IBM1124,	8 位：西里爾文，白俄羅斯
1140	IBM01140, IBM -1140,	主辦單位：USA、加拿大 (ESA)、荷蘭、葡萄牙、巴西、澳洲、紐西蘭、歐元

CCSID	字元集	描述
1141	IBM1141, IBM -1141	主辦單位：奧地利、德國、歐元
1142	IBM1142, IBM -1142,	主辦單位：丹麥、挪威、歐元
1143	IBM1143, IBM -1143,	主辦單位：芬蘭、瑞典、歐元
1144	IBM1144, IBM -1144,	主辦單位：意大利，歐元
1145	IBM一百四IBM十五	主辦單位：西班牙、拉丁美洲（西班牙）、歐元
1146	IBM1146, IBM -1146	主機：英國，歐元
1147	IBM1147, IBM -1147	主辦單位：法國，歐元
1148	IBM1148, IBM -1148,	主辦機構：比利時、加拿大 (AS/400)、瑞士、國際拉丁 1、歐元
1149	IBM1149, IBM -1149,	主辦單位：冰島，歐元
1200	UTF-16 是	包含字元集 65535 的統一碼。在沒有字節順序標記（BOM）的情況下，假定為 UTF -16 BE（大端）。
1202	UTF-16 勒	UTF-16 勒同一个 IBM PUA
1204	UTF-16	UTF-16 與 IBM PUA
1208	UTF-8, UTF -8 焦耳, UTF8	包含字元集 65535 的統一碼。UTF-8.
1232	UTF-32	UTF-32 是與 IBM PUA
1234	UTF-32 勒	UTF-32 勒同 IBM PUA
1236	UTF-32	UTF-32 與 IBM PUA

CCSID	字元集	描述
1351	IBM-1351	JAPAN OPEN
1362	IBM-1362	KOREAN女士 WIN
1363	IBM視IBM窗 MS949	電腦數據：MS 視窗韓國 SBCS。PC 數據:MS 視窗古蘭經DBCS包括 11172 全韓文
1364	IBM-1364	主機：已延伸SBCS。主機：DBCS包括 1880 個使用者定義的字元和 11172 個完整的韓文字元
1370	IBM-1370	PC 數據：擴展SBCS，歐元。PC 數據：DBCS包括 6204 個用戶定義的字符，歐元
1371	IBM-1371	主持人：擴展SBCS，歐元。主機：DBCS包括 6204 個使用者定義的字元，含歐元
1375	大五 HKSCS	混合大 -5 分機 HKSCS
1380	IBM-1380	個人電腦資料：S-CH GB
1381	IBM-1381, X-IBM1381,	電腦資料：延伸 SBCS (IBMGB)。電腦資料：DBCS(IBMGB)，包括 31 個IBM已選取的、1880 個使用者定義的字元
1382	IBM-1382	S-CHINESE EUC
1383	EUC-中国,GB2312, IBM -1383, X-, IBM1383	G0:ASCII. 一級：國標 2312-80 集
1385	IBM-1385	電腦資料：S-CH GBK

CCSID	字元集	描述
1386	GBK、、IBM視窗、MS936	電腦數據：S 中文GBK和 T-中文 -5 IBM BIG。電腦資料：S-中文 GBK
1388	IBM-1388	主機：已延伸SBCS。主機：DBCS包括 1880 個使用者定義字元
1390	IBM-1390	片假名主持人：延長SBCS，歐元。漢字主機：DBCS包含 6205 個使用者定義的字元
1399	IBM-1399	拉丁主機：擴展SBCS，歐元。漢字主機：DBCS包括 4370 個用戶定義的字符，歐元
5050	JIS0201、JIS 0208、JIS 0212、0 JIS 201、0208、0212 JIS JIS	一級JIS賽：羅馬。一級：JISX208-1990。一級：片JIS假名。國一 JIS
5054	ISO-2022 日本	JAPANESE TCP
5346	視窗	MS 視窗:拉丁 -2, 版本 2 與歐元
5347	視窗	MS 視窗：西里爾文，版本 2 與歐元
5348	視窗	MS 視窗:拉丁 1 國家, 版本 2 與歐元
5349	視窗	MS 視窗：希臘，版本 2 與歐元
5350	視窗	MS 視窗:土耳其, 版本 2 與歐元

CCSID	字元集	描述
5351	視窗	MS 視窗：希伯來語，版本 2 與歐元
5352	視窗 -1256, 視窗	MS 視窗:阿拉伯語, 版本 2 與歐元
5353	視窗	MS 視窗：波羅的海邊緣，版本 2 與歐元
5354	視窗	MS 視窗：越南語，版本 2 與歐元
5488	GB18030	GB18030、1 位元組資料 GB18 030、2 位元組資料 030、4 位元組資料 GB18
9030	IBM-838	主持人：泰國擴展 SBCS
9066	IBM-874	PC 數據：泰語擴展 SBCS
9400	CESU-8	CESU-8 與 IBM PUA
25546	ISO-2022-克朗	KOREAN TCP
33722	IBM-33722, -33722C IBM	IBMeucJP

AWS 大型主機現代化應用程式測試中的資料保護

共同責任模型 AWS 共同責任模型適用於大型 AWS 主機現代化應用程式測試中的資料保護。如此模型中所述， AWS 負責保護執行所有 AWS 雲端. 您負責維護在此基礎設施上託管內容的控制權。您也同時負責所使用 AWS 服務 的安全組態和管理任務。如需資料隱私權的詳細資訊，請參閱資料隱私權常見問答集。如需有關歐洲資料保護的相關資訊，請參閱 AWS 安全性部落格上的 AWS 共同的責任模型和 GDPR 部落格文章。

建議您使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 保護 AWS 帳戶 登入資料並設定個別使用者。因此，每個使用者只會獲得履行其工作職責所需的權限。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源進行通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 使用設定 API 和使用者活動記錄 AWS CloudTrail。
- 使用 AWS 加密解決方案以及其中的所有默認安全控制 AWS 服務。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Amazon S3 的敏感資料。
- 如果您在透過命令列介面或 API 存取時需要經 AWS 過 FIPS 140-2 驗證的加密模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的更多相關資訊，請參閱聯邦資訊處理標準 (FIPS) 140-2 概觀。

建議您避免將任何機密或敏感資訊 (例如客戶的電子郵件地址) 用於標籤或任意格式的文字欄位 (例如「名稱」欄位)。這包括當您使用主控台、API 或 SDK 使用 AWS 大型主機現代化應 AWS 服務 用程式測試或其他時間。 AWS CLI AWS 您在用於名稱的標籤或任意格式文字欄位中輸入的任何資料，都可能會用於帳單或診斷記錄。如果您向外部伺服器提供 URL，請避免使用 URL 中的認證資訊來驗證您對該伺服器的要求。

AWS 大型主機現代化應用程式測試所收集的資料

AWS 大型主機現代化應用程式測試會從您那裡收集數種類型的資料：

- `Resource definition`：資源定義指示當您建立或更新測試案例、測試套件或測試組態類型的資源時，傳送至「應用程式測試」的資料。
- `Scripts for replay`：這些是針對您的 AWS 大型主機現代化應用程式傳遞至應用程式測試的指令碼。
- `Data for comparison`：這些是傳遞給應用程序測試進行比較的數據集或數據庫更改數據捕獲（CDC）文件。

AWS 大型主機現代化應用程式測試將此資料原生儲存在. AWS AWS 我們向您收集的資料存放在大型主機現代化應用程式測試管理的 Amazon S3 儲存貯體中。刪除資源時，相關聯的資料會從 Amazon S3 儲存貯體中移除。

當您開始測試回合以執行重播以測試互動式工作負載時， AWS 大型主機現代化應用程式測試會將指令碼下載到暫時儲存備份-Amazon ECS 託管的 Fargate 容器中，以執行重新執行。重播完成後，指令碼檔案就會刪除，並將指令碼產生的輸出檔案存放在您帳戶的應用程式測試管理的 Amazon S3 儲存貯體中。刪除測試回合時，重播輸出檔案會從 Amazon S3 儲存貯體中刪除。

同樣地，當您開始測試回合以比較檔案 (資料集或資料庫變更) 時， AWS 大型主機現代化應用程式測試會將檔案下載到暫時儲存 Backked-Amazon ECS 託管的 Fargate 容器中，以執行比較。一旦比較操

作完成，下載的文件將被刪除。比較輸出資料會存放在您帳戶的應用程式測試管理的 Amazon S3 儲存貯體中。刪除測試回合時，會從 S3 儲存貯體刪除輸出資料。

當您將資料放置在 AWS 大型主機現代化應用程式測試用來比較檔案的 Amazon S3 儲存貯體時，您可以使用所有可用的 Amazon S3 加密選項來保護資料。

AWS 大型主機現代化應用程式測試的靜態資料加密

AWS 大型主機現代化應用程式測試與 AWS Key Management Service (KMS) 整合，可在永久儲存資料的所有相依資源上提供透明的伺服器端加密 (SSE)。資源範例包括 Amazon 簡單儲存服務、Amazon DynamoDB 和亞馬遜彈性區塊存放區。 AWS 大型主機現代化應用程式測試會在中為您建立和管理對稱加密 AWS KMS 金鑰。 AWS KMS

依預設加密靜態資料，有助於降低保護敏感資料所涉及的營運開銷和複雜性。同時，它可讓您測試需要嚴格加密合規性和法規要求的應用程式。

當您建立測試案例、測試套件或測試設定時，您無法停用此層加密或選取替代加密類型。

您可以使用自己的客戶受管金鑰來比較檔案和 AWS CloudFormation 範本，以加密 Amazon S3。您可以使用此密鑰來加密應用程序測試中為測試運行創建的所有資源。

> **Note**
>
> DynamoDB 資源一律使用應用程式測試服務帳戶 AWS 受管金鑰 中的加密。您無法使用客戶受管金鑰加密 DynamoDB 資源。

AWS 大型主機現代化應用程式測試會使用客戶管理的金鑰來執行下列工作：

- 將資料集從應用程式測試匯出到 Amazon S3。
- 將比較輸出檔案上傳到 Amazon S3。

如需更多資訊，請參閱 AWS Key Management Service 開發人員指南中的客戶受管金鑰。

建立客戶受管金鑰

您可以使用 AWS Management Console 或 AWS KMS API 建立對稱的客戶管理金鑰。

建立對稱客戶受管金鑰

請依照《AWS Key Management Service 開發人員指南》中建立對稱客戶受管金鑰的步驟進行。

金鑰政策

金鑰政策會控制客戶受管金鑰的存取權限。每個客戶受管金鑰都必須只有一個金鑰政策，其中包含決定誰可以使用金鑰及其使用方式的陳述式。在建立客戶受管金鑰時，可以指定金鑰政策。

以下是範例金鑰原則範圍縮減存取範圍，可讓應用程式測試在您的帳戶中寫入重播和比較產生的資料。ViaService 呼叫 `StartTestRun` API 時，應將此政策附加至 IAM 角色。

Example

```
{
    "Sid": "TestRunKmsPolicy",
    "Action": ["kms:Decrypt", "kms:GenerateDataKey"],
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/TestRunRole"
    },
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "kms:ViaService": ["s3.amazonaws.com"]
        },
        "ForAnyValue:StringEquals": {
            "kms:EncryptionContextKeys": "aws:apptest:testrun"
        }
    }
}
```

如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的管理客戶受管金鑰的存取。

如需有關故障診斷金鑰存取的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》。

為 AWS 大型主機現代化應用程式測試指定客戶管理的金鑰

建立測試組態時，您可以透過輸入 KEY ID 來指定客戶管理的金鑰。應用程式測試用於在測試執行期間加密上傳到 Amazon S3 儲存貯體的資料。

- 金鑰 ID — 客戶管理金鑰的金鑰識別碼。輸入金鑰 ID、金鑰 ARN、別名名稱或別名 ARN。

若要在使用建立測試組態時新增客戶管理的金鑰 AWS CLI，請指定kmsKeyId參數，如下所示：

```
create-test-configuration --name test \
--resources '[{
    "name": "TestApplication",
    "type": {
        "m2ManagedApplication": {
            "applicationId": "wqju4m2dcz3rhny5fpdozrsdd4",
            "runtime": "MicroFocus"
        }
    }
}]' \
--service-settings '{
    "kmsKeyId": "arn:aws:kms:us-west-2:111122223333:key/05d467z6-c42d-40ad-
b4b7-274e68b14013"
}'
```

AWS 大型主機現代化應用程式測試加密內容

加密內容是一組選用的金鑰值對，包含資料的其他相關內容資訊。

AWS KMS 使用加密內容作為其他驗證資料，以支援已驗證的加密。當您在加密資料的要求中包含加密內容時，會將加密內容 AWS KMS 繫結至加密的資料。若要解密資料，您必須在請求中包含相同的加密內容。

AWS 大型主機現代化應用程式測試加密內容

AWS 大型主機現代化應用程式測試在與測試回合相關的所有 AWS KMS 密碼編譯作業中使用相同的加密內容，其中索引鍵是，aws:apptest:testrun而值是測試回合的唯一識別碼。

Example

```
"encryptionContext": {
        "aws:apptest:testrun": "u3qd7uhdandgdkhhi44qv77iwq"
}
```

使用加密內容進行監控

當您使用對稱的客戶受管金鑰來加密測試執行時，您也可以在稽核記錄和日誌中使用加密內容，以識別將資料上傳到 Amazon S3 時客戶受管金鑰的使用方式。

監控 AWS 大型主機現代化應用程式測試的加密金鑰

當您將 AWS KMS 客戶受管金鑰與 AWS 大型主機現代化應用程式測試資源搭配使用時，您可以使用AWS CloudTrail追蹤 AWS 大型主機現代化應用程式測試在上傳物件時傳送給 Amazon S3 的請求。

傳輸中加密

對於定義測試交易工作負載步驟的測試案例，應用程式測試管理終端模擬器執行硒指令碼和 AWS 大型主機現代化應用程式端點之間的資料交換不會在傳輸過程中加密。 AWS 大型主機現代化應用程式測試使用連線 AWS PrivateLink 到您的應用程式端點，以私密交換資料，而不會透過公用網際網路暴露流量。

AWS 大型主機現代化應用程式測試使用 HTTPS 加密服務 API。 AWS 大型主機現代化應用程式測試中的所有其他通訊都受到服務 VPC 或安全群組以及 HTTPS 的保護。

依預設會設定傳輸中的基本加密，但不適用於以TN3270通訊協定為基礎的互動式工作負載測試。

AWS 大型主機現代化中的檔案傳輸

AWS大型主機現代化檔案傳輸可讓您將大型主機資料集傳輸和轉換為 Amazon S3，以進行大型主機現代化、遷移和擴充使用案例。它簡化了將資料集從大型主機傳輸到. AWS 雲端主要功能包括：探索來源大型主機資料集和成品，以及可加快資料傳輸至 Amazon S3 的可擴展性和效率。檔案傳輸支援各種大型主機資料集類型，例如順序、PDS、GDSGDG、和。VSAM KSDS此服務會將資料集傳輸到中繼 Amazon S3 儲存貯體，將它們轉換為指定的目標代碼頁，然後將它們移至所需的目標 S3 儲存貯體。

主題

- 什麼是AWS大型主機現代化檔案傳輸？
- 安裝檔案傳輸代理程式
- 設定檔案傳輸代理程式
- 建立檔案傳輸的資料傳輸端點
- 在「檔案傳輸」中建立傳送任
- 教學課程：開始使用AWS大型主機現代化檔案傳輸
- AWS大型主機現代化檔案傳輸中支援的來源與目標編碼

什麼是AWS大型主機現代化檔案傳輸？

使用AWS大型主機現代化檔案傳輸，您可以使用全受管服務傳輸和轉換資料集和檔案，以加速並簡化 AWS 大型主機現代化服務和 Amazon S3 的現代化、遷移和增強使用案例。

主題

- AWS大型主機現代化檔案傳輸的優點
- AWS大型主機現代化檔案傳輸的運作方式

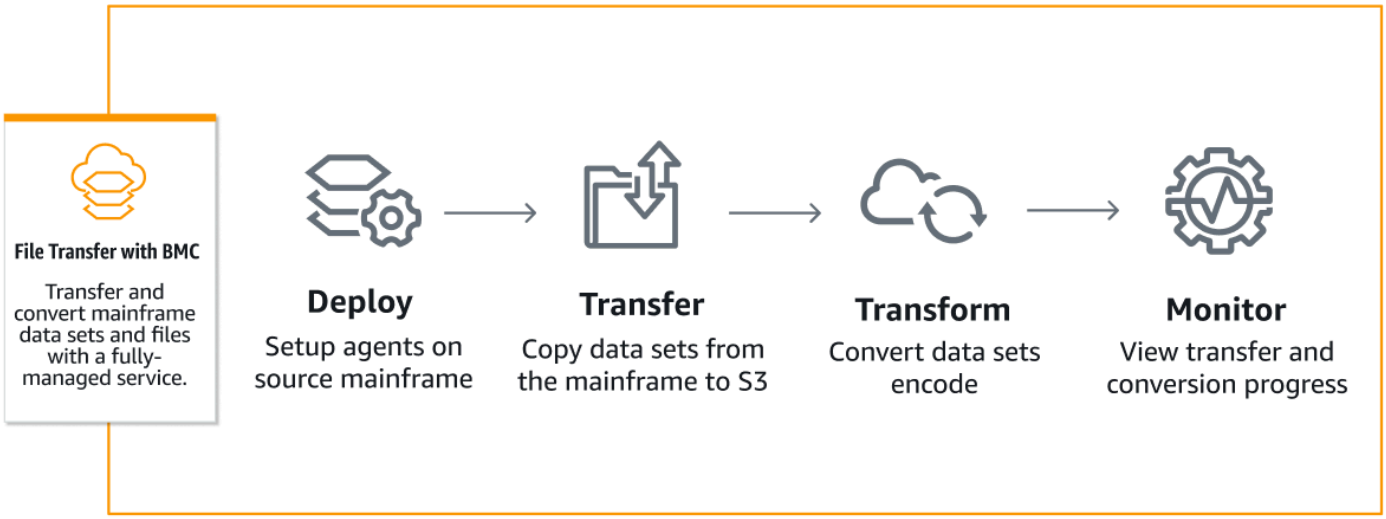
AWS大型主機現代化檔案傳輸的優點

AWS大型主機現代化檔案傳輸可協助您將資料集從大型主機傳輸到 Amazon S3。一些好處包括：

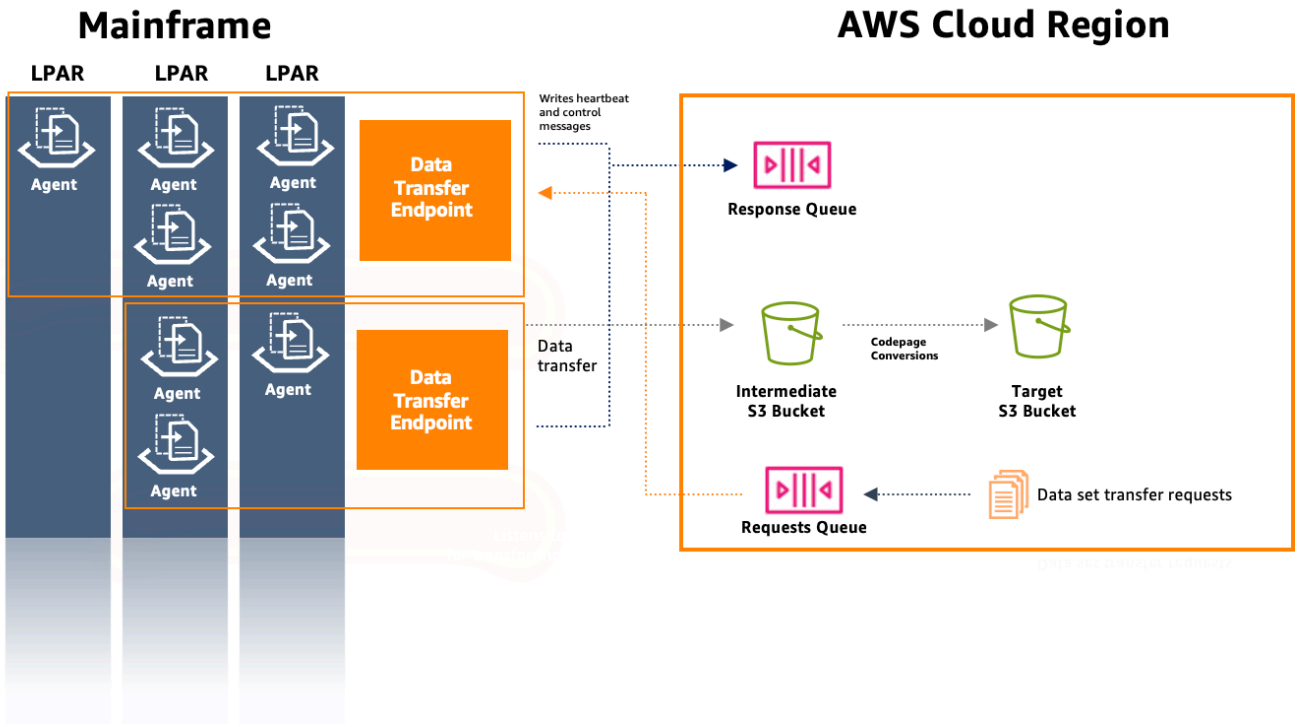
- 探索來源大型主機資料集和成品
- 自動傳輸和資料集轉換
- 可擴展性，效率和速度，實現更快的數據集傳輸 AWS

AWS大型主機現代化檔案傳輸的運作方式

下圖概述AWS大型主機現代化檔案傳輸在概念層面上的運作方式。



下圖是大型主機現代化檔案傳輸功能AWS的架構概觀。



安裝檔案傳輸代理程式

您可以使用本文件做為在來源大型主機上安裝代理程式的 step-by-step 指南。

主題

- 步驟 1：為 M2 代理程式建立 ZF 資料集
- 步驟 2：將資料集格式化為 ZF
- 步驟 3：掛載檔案系統
- 步驟 4：驗證掛載
- 步驟 5：輸入 OMVS
- 步驟 6：設定代理程式安裝目錄環境變數
- 步驟 7：設定工作目錄環境變數
- 步驟 8：建立工作目錄
- 步驟 9：複製代理程式 tar 檔案並複製工作目錄
- 步驟 10：假設根使用者
- 步驟 11：完成代理程式安裝

步驟 1：為 M2 代理程式建立 ZF 資料集

使用下列指令建立 M2 代理程式安裝的 ZF。JCL

```
//DEFINE   EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
 DEFINE CLUSTER (NAME(yourhlq.M2AGENT.ZFS) -
 VOLUMES(*) -
 LINEAR CYL(1000 200))
```

步驟 2：將資料集格式化為 ZF

建立資料集之後，將其格式化為 ZF 檔案系統。

其中一種方法是使用下列 Job 控制語言 (JCL)：

```
//FORMAT  EXEC PGM=IOEAGFMT,PARM='AGGRNAME(yourhlq.M2AGENT.ZFS),FORMAT,AGGRSIZE(1200)'
//SYSPRINT DD  SYSOUT=A
```

提交此工作並檢查是否成功完成。

步驟 3：掛載檔案系統

若要掛載檔案系統，請使用MOUNT指令。您可以在命令行中ISPF或批量掛載文件系統。

例如：

```
MOUNT FILESYSTEM('yourhlq.M2AGENT.ZFS') TYPE(ZFS) MODE(RDWR) MOUNTPOINT('/usr/lpp/aws/
m2-agent')
```

步驟 4：驗證掛載

使用D OMVS,F命令或在 Unix 系統服務 (USS) 內進行檢查，驗證檔案系統是否已正確掛載。

步驟 5：輸入 OMVS

使用下列指令輸入OMVS：

```
TSO OMVS
```

步驟 6：設定代理程式安裝目錄環境變數

使用下列命令來設定代理程式安裝目錄環境：

```
export AGENT_DIR=/usr/lpp/aws/m2-agent
```

> **Note**
>
> 掛接點是在步驟 3 中定義的。

步驟 7：設定工作目錄環境變數

使用下列指令來設定工作目錄環境變數：

```
export WORK_DIR=$AGENT_DIR/tmp
```

步驟 8：建立工作目錄

使用下列命令來設定工作目錄環境：

```
mkdir -p $WORK_DIR
```

步驟 9：複製代理程式 tar 檔案並複製工作目錄

從AWS使用 M2 代理程式連結下載代理程式 tar 檔案。

傳輸機制將取決於您的環境，但請確保 tar 文件以二進制模式傳輸。

步驟 10：假設根使用者

使用下列命令假設 root 使用者：

```
su
```

步驟 11：完成代理程式安裝

請依照下列步驟完成代理程式安裝。

1. 使用下列指令將 m2-agent 版本環境變數設定為目前正在安裝的版本：

   ```
   export M2_AGENT_VERSION=1.0.0
   ```

2. 使用下列命令解壓縮代理程式 tar 套件：

   ```
   tar -xpf m2-agent-package-$M2_AGENT_VERSION.tar -C $AGENT_DIR
   ```

3. 使用下列命令建立目前代理程式安裝目錄的`current-version`符號連結：

   ```
   ln -s $AGENT_DIR/m2-agent-v$M2_AGENT_VERSION $AGENT_DIR/current-version
   ```

4. 更新並送出CPY#PDS以建立檔案傳輸代理程式資料集。

   > **Note**
   >
   > JCL使用SYS2.AWS.M2 HLQ.

若要建立檔案傳輸代理程式，請設定參數行 000006-000012。此外，AGNTPATH請更新三個符號變數HLQVOLSER、和，稍後在下列項目中使用JCL：

```
oedit $AGENT_DIR/current-version/installation/CPY#PDS
submit $AGENT_DIR/current-version/installation/CPY#PDS
```

> **Note**
>
> 這JCL是為了在大型主機上設定代理程式安裝的特定層面而量身打造。它分配必要的數據集，然後從 Unix 文件系統複製特定文件到這些數據集。

設定檔案傳輸代理程式

安裝檔案傳輸代理程式之後，請依照下列步驟設定代理程式。如果您需要安裝新的代理程式，請遵循the section called “安裝檔案傳輸代理程式”頁面上的指示。

主題

- 步驟 1：配置權限和啟動任務控制（STC）
- 步驟 2：創建 Amazon S3 存儲桶
- 步驟 3：建立用於加密的 AWS KMS 客戶管理金鑰
- 步驟 4：建立大型主機認證的 AWS Secrets Manager 密碼
- 步驟 5：建立IAM策略
- 步驟 6：建立具有長期存取認證的IAM使用者
- 步驟 7：建立代理程式要承擔的IAM角色
- 步驟 8：代理程式組態

步驟 1：配置權限和啟動任務控制（STC）

1. 根據其指示更新並提交其中一個SYS2.AWS.M2.SAMPLIB(SEC#TSS)（用於設置TSS權限）或（用於設置權限）。SYS2.AWS.M2.SAMPLIB(SEC#RACF) RACF這些成員是由上一個CPY#PDS步驟建立的。

> **Note**
>
> SYS2.AWS.M2是安裝期間選擇的高階限定詞 (HLQ)。

2. 如果預設的檔案傳輸代理程式目錄路徑 (/usr/lpp/aws/m2-agent) 已變更 SYS2.AWS.M2.SAMPLIB(M2AGENT) STCJCL，請在中更新PWD匯出。
3. 更新並複製SYS2.AWS.M2.SAMPLIB(M2AGENT)JCL到SYS1.PROCLIB。
4. 使用下APF列指令新增SYS2.AWS.M2.LOADLIB至清單：

```
SETPROG APF ADD DSNAME(SYS2.AWS.M2.LOADLIB) SMS
```

5. 將代理logs和diag文件夾的組和所有者設置為代理用戶/組（ M2 /M2USER ）。GROUP使用下列命令：

```
chown -R M2USER:M2GROUP $AGENT_DIR/current-version/logs
chown -R M2USER:M2GROUP $AGENT_DIR/current-version/diag
```

步驟 2：創建 Amazon S3 存儲桶

AWS大型主機現代化檔案傳輸需要將中繼 Amazon S3 儲存貯體做為工作區。我們建議您專門為此建立值區。

或者，為傳輸的資料集建立新的目標 Amazon S3 儲存貯體。否則，您也可以使用現有的 Amazon S3 儲存貯體。如需建立 Amazon S3 儲存貯體的詳細資訊，請參閱建立儲存貯體。

步驟 3：建立用於加密的 AWS KMS 客戶管理金鑰

若要在中建立客戶管理的金鑰 AWS KMS

1. 在開啟 AWS KMS 主控台https://console.aws.amazon.com/kms。
2. 在左側導覽窗格中選擇 [客戶管理金鑰]。
3. 選擇建立金鑰。
4. 在 [設定金鑰] 下，選擇 [金鑰類型] 為 [對稱] 和 [金鑰使用] 做為加密和解密。使用其他預設組態。
5. 在 「加入標示」中，為您的金鑰加入別名和描述。
6. 選擇 Next (下一步)。

7. 在 [定義金鑰管理權限] 下，至少選擇一個管理此金鑰的IAM使用者和角色。
8. 選擇 Next (下一步)。
9. 在 [複查] 頁面上，將下列語法新增至金鑰原則。這可 AWS 讓大型主機現代化服務讀取並使用這些金鑰進行加密/解密。

> **Important**
>
> 將陳述式新增至現有陳述式。請勿取代原則中已有的內容。

```
{
    "Sid" : "Enable AWS M2 File Transfer Permissions",
    "Effect" : "Allow",
    "Principal" : {
        "Service" : "m2.amazonaws.com"
    },
    "Action" : [
        "kms:Encrypt",
        "kms:Decrypt"
    ],
   "Resource" : "*"
},
```

建立客戶管理的金鑰後，儲存該金鑰。ARN它將在稍後的策略中使用。

步驟 4：建立大型主機認證的 AWS Secrets Manager 密碼

需要大型主機認證才能存取要傳輸的資料集，而且這些認證必須儲存為 AWS Secrets Manager 密碼。

若要建立 AWS Secrets Manager 密碼

1. 開啟密碼管理員主控台的位https://console.aws.amazon.com/secretsmanager置
2. 在 [選擇密碼類型] 中，選擇 [其他密碼類型]。
3. 使用 userId 可存取資料集之大型主機的索引鍵值`userId`。
4. 使用密碼欄位`password`的金鑰值。
5. 在「加密金鑰」中，選擇先前建立的 AWS 客戶管理金鑰。
6. 選擇 Next (下一步)。

7. 在 [設定密碼] 頁面上，提供名稱和說明。
8. 在相同的頁面上，編輯資源權限，並使用下列資源原則， AWS 讓大型主機現代化服務可以存取它。

```
{
  "Version" : "2012-10-17",
  "Statement" : [ {
    "Effect" : "Allow",
    "Principal" : {
        "Service" : "m2.amazonaws.com"
    },
    "Action" : [ "secretsmanager:GetSecretValue",
                 "secretsmanager:DescribeSecret" ],
    "Resource" : "*"
  } ]
}
```

9. 選擇 [儲存] 儲存更新的權限，然後再選擇 [下一步
10. 略過「設定自動重建」頁面，然後選擇「下一步」。
11. 在 [檢閱] 頁面上，檢查所有組態，然後選擇 [儲存] 以儲存密碼。

> **Important**
>
> `userId`和`password`秘密金鑰區分大小寫，必須如圖所示輸入。

步驟 5：建立IAM策略

使用代理程式所需權限建立新原則

1. 從可視化編輯器切換到JSON編輯器，並以下列範本取代內容：

```
{
"Version": "2012-10-17",
    "Statement": [
        {
         "Sid": "FileTransferAgentSQSReceive",
            "Effect": "Allow",
            "Action": [
                "sqs:DeleteMessage",
```

```
                "sqs:ReceiveMessage"
            ],
            "Resource": "arn:aws:sqs:*:111122223333:m2-*-request-queue.fifo"
        },
        {
         "Sid": "FileTransferAgentSQSSend",
            "Effect": "Allow",
            "Action": "sqs:SendMessage",
            "Resource": "arn:aws:sqs:*:111122223333:m2-*-response-queue.fifo"
        },
        {
         "Sid": "FileTransferWorkingS3",
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "<file-transfer-endpoint-intermediate-bucket-arn>/*"
        },
        {
         "Sid": "FileTransferAgentKMSDecrypt",
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "<kms-key-arn>"
        }
    ]
}
```

2. 將請求隊列和響應隊列111122223333中的替換為您ARN的帳戶。

> **Note**
>
> 這些ARN是萬用字元，與資料傳輸端點初始化期間建立的兩個 Amazon SQS 佇列相符。創建文件傳輸端點後，可以選擇將ARN這些端點替換為來自 Amazon 的實際值SQS。

3. 取代`file-transfer-endpoint-intermediate-bucket-arn`為先前建立ARN的移轉時段。保留「/*」通配符在末尾。
4. `kms-key-arn`以先前建立ARN的 AWS KMS 金鑰取代。

步驟 6：建立具有長期存取認證的IAM使用者

建立IAM允許大型主機代理程式連線到您帳戶的 AWS 使用者。代理程式將與此使用者連線，然後假設您定義的角色具有使用 Amazon SQS 回應和請求佇列的權限，以及將資料集儲存到 Amazon S3 儲存貯體。

若要建立此IAM使用者

1. 導航到AWSIAM控制台https://console.aws.amazon.com/iam。
2. 在 [權限] 選項中，選擇 [直接連結原則] 選項，但不附加任何權限原則。這些權限將由將附加的角色管理。
3. 創建用戶後，選擇用戶並打開安全憑據選項卡。
4. 在 [建立存取金鑰] 中，當系統提示輸入 [使用案例] 時選擇
5. 複製並安全地保存生成的訪問密鑰和秘密訪問密鑰。這些將在以後使用。

如需有關建立IAM存取金鑰的詳細資訊，請參閱管理IAM使用者的存取金鑰。

> **Important**
>
> 在選擇「完成」之前，先儲存存取金鑰建立精靈最後一頁上顯示的「存取金鑰」和「密碼」存取金鑰。這些金鑰可用來設定大型主機代理程式。

> **Note**
>
> 儲存IAMARN用來設定與角色之信任關係的使用IAM者。

步驟 7：建立代理程式要承擔的IAM角色

建立代理程式的新IAM角色

1. 在IAM主控台中選擇 [角色] https://console.aws.amazon.com/iam。
2. 選擇建立角色。
3. 在 [選取信任的實體] 頁面上，為 [信任的實體類型] 選擇 [自訂信任原則]。
4. 以下列項目取代 [自訂信任原則]，並取代<iam-user-arn>為之前建立ARN的使用者。

```
{
    "Version": "2012-10-17",
    "Statement": [ {
        "Sid": "FileTransferAgent",
        "Effect": "Allow",
        "Principal": {
```

```
            "AWS": "<IAM-User-arn>"
        },
        "Action": "sts:AssumeRole"
    } ]
}
```

5. 選擇 Next (下一步)。
6. 在 [新增權限] 中，篩選您先前建立的原則名稱並加以選擇。
7. 選擇 Next (下一步)。
8. 為角色命名，然後選擇建立角色。

> **Note**
>
> 儲存角色名稱，您稍後將使用此名稱來設定大型主機代理程式。

步驟 8：代理程式組態

設定檔案傳輸代理程式

1. 導覽至 `$AGENT_DIR/current-version/config`。
2. 使用下列命令編輯代理程式的組態檔`appication.properties`，以新增環境組態：

```
oedit $AGENT_DIR/current-version/config/application.properties
```

例如：

```
agent.environments[0].account-id=<AWS_ACCOUNT_ID>
agent.environments[0].agent-role-name=<AWS_IAM_ROLE_NAME>
agent.environments[0].access-key-id=<AWS_IAM_ROLE_ACCESS_KEY>
agent.environments[0].secret-access-id=<AWS_IAM_ROLE_SECRET_KEY>
agent.environments[0].bucket-name=<AWS_S3_BUCKET_NAME>
agent.environments[0].environment-name=<AWS_REGION>
agent.environments[0].region=<AWS_REGION>
zos.complex-name=<File_Transfer_Endpoint_Name>
```

其中：

- `AWS_ACCOUNT_ID`是帳戶的識別 AWS 碼。

- AWS_IAM_ROLE_NAME是在中建立之IAM角色的名稱the section called “步驟 7：建立代理程式要承擔的IAM角色”。
- AWS_IAM_ROLE_ACCESS_KEY是在中建立之IAM使用者的存取金鑰the section called “步驟 6：建立具有長期存取認證的IAM使用者”。
- AWS_IAM_ROLE_SECRET_KEY是在中建立之IAM使用者的存取密鑰the section called “步驟 6：建立具有長期存取認證的IAM使用者”。
- AWS_S3_BUCKET_NAME是使用資料傳輸端點建立的傳輸值區的名稱。
- AWS_REGION是您設定檔案傳輸代理程式的區域。

> **Note**
>
> 您可以 AWS 透過定義多個環境，將檔案傳輸代理程式移轉至中的多個區域和帳戶。

- (選擇性)。 zos.complex-name是您在建立檔案傳輸端點時建立的複雜名稱。

> **Note**
>
> 只有當您要自訂與建立檔案傳輸端點時所定義的相同複雜名稱 (預設為您的 sysplex 名稱) 時，才需要此欄位。如需詳細資訊，請參閱the section called “建立資料傳輸端點”。

> **Important**
>
> 可以有幾個這樣的部分，只要括號中的索引 — [0] — 為每個部分遞增。

您必須重新啟動代理程式，變更才會生效。

需求

1. 新增或移除參數時，代理程式必須停止並啟動。使用中的下列命令啟動檔案傳輸代理程式CLI：

```
/S M2AGENT
```

若要停止 M2 代理程式，請在中使用下列指令CLI：

```
/P M2AGENT
```

2. 您可以 AWS 透過定義多個環境，將檔案傳輸代理程式移轉至中的多個區域和帳戶。

> **Note**
>
> 使用您先前建立並設定的參數值取代這些值。

```
#Region 1
agent.environments[0].account-id=AWS_ACCOUNT_ID
agent.environments[0].agent-role-name=AWS_IAM_ROLE_NAME
agent.environments[0].access-key-id=AWS_IAM_ROLE_ACCESS_KEY
agent.environments[0].secret-access-id=AWS_IAM_ROLE_SECRET_KEY
agent.environments[0].bucket-name=AWS_S3_BUCKET_NAME
agent.environments[0].environment-name=AWS_REGION
agent.environments[0].region=AWS_REGION

#Region 2
agent.environments[1].account-id=AWS_ACCOUNT_ID
agent.environments[1].agent-role-name=AWS_IAM_ROLE_NAME
agent.environments[1].access-key-id=AWS_IAM_ROLE_ACCESS_KEY
agent.environments[1].secret-access-id=AWS_IAM_ROLE_SECRET_KEY
agent.environments[1].bucket-name=AWS_S3_BUCKET_NAME
agent.environments[1].environment-name=AWS_REGION
agent.environments[1].region=AWS_REGION
```

建立檔案傳輸的資料傳輸端點

資料傳輸端點可實現與來源大型主機的連線能力，並支援高可用性、延展性和簡化的代理程式管理。個別代理程式會安裝在大型主機上，LPARs而且可以一起分組成資料傳輸端點。當要求傳輸資料集時，資料傳輸端點中的一個代理程式會處理該特定傳輸。若要起始資料傳輸，資料傳輸端點上的至少一個代理程式必須處於線上狀態。

此程序假設您已完成來源大型主機上設定 AWS 大型主機現代化設定檔案傳輸代理程式中的步驟。

建立資料傳輸端點

若要建立檔案傳輸的資料傳輸端點，請依照大型主機現代化 AWS 主控台中的下列步驟執行。

建立資料傳輸端點

1. 在開啟大 AWS 型主機現代化主控台。https://console.aws.amazon.com/m2/
2. 在選擇 AWS 區域 器中，選擇您要將檔案從大型主機傳輸到 Amazon S3 儲存貯體的區域。
3. 在「資料傳輸端點」頁面的「檔案傳輸」下，選擇「建立資料傳輸端點」。
4. 在 [資料傳輸端點必要條件] 頁面上，閱讀所有指示，確定您已在來源大型主機上完成這些步驟。確認後，選擇「下一步」。
5. 在 [設定資料傳輸端點] 頁面上，新增資料傳輸端點的基本資訊。

    1. 在「基本資訊」區段中，輸入您的資料傳輸端點名稱。

    > **Note**
    >
    > 除非您在代理程式組態中指定複雜名稱，否則資料傳輸端點名稱必須與 Sysplex 名稱相符。

    2. 選擇性的描述。
    3. 用來加密密KMS碼的金鑰。

    > **Note**
    >
    > 您必須新增下列以資源為基礎的原則KMS， AWS 大型主機現代化服務才能讀取並使用這些金鑰進行加密/解密：
    >
    > ```
    > {
    >   "Sid" : "Enable AWS M2 Permissions",
    >   "Effect" : "Allow",
    >   "Principal" : {
    >       "Service" : "m2.amazonaws.com"
    >             },
    >    "Action" : [
    >        "kms:Encrypt",
    >        "kms:Decrypt"
    >               ],
    >    "Resource" : "*"
    > }
    > ```

    4. 指定中繼資料的 S3 位置，這是從大型主機傳輸的資料集在轉換並傳輸到目標 Amazon S3 儲存貯體之前存放在這些資料集的中繼 S3 位置。

.

> **Note**
>
> 建議您為傳輸任務建立新的 Amazon S3 儲存貯體。如需詳細資訊，請參閱建立值區。您也可以選擇瀏覽 S3 選項，瀏覽現有的 Amazon S3 儲存貯體。

5. 輸入必填欄位後，選擇 [下一步]。
6. 在 [檢閱並建立資料傳輸端點] 頁面上，檢查是否已完成必要條件，然後檢閱基本資訊。確認後，選擇 [建立資料傳輸端點]。

系統會將您重新導向至「資料傳輸端點概觀」頁面，您可以在此查看所有資料傳輸端點的清單。您也可以看到可用或已失敗的資料傳輸端點。

您也可以依名稱搜尋資料傳輸端點，並存取每個可用代理程式的其他資訊。

在「檔案傳輸」中建立傳送任

傳輸任務用於指定要從大型主機傳輸到 Amazon S3 的資料集，並可讓您選擇字碼頁轉換選項。

這些指示假設您已完成中的步驟設定 AWS 大型主機現代化並已建立the section called “建立資料傳輸端點”。

主題

- 建立移轉任務
- 檢視移轉任務

建立移轉任務

若要在檔案傳輸中建立傳輸工作，請依照大型主機現代化 AWS 主控台中的下列步驟執行。

若要建立移轉任務

> **Important**
>
> 您必須至少有一個資料傳輸端點，才能建立新的傳輸工作。

1. 在開啟大 AWS 型主機現代化主控台。https://console.aws.amazon.com/m2/
2. 在選擇 AWS 區域 器中，選擇您要將檔案從大型主機傳輸到 Amazon S3 儲存貯體的區域。
3. 在 [傳送任務] 頁面上，您可以選擇任何資料傳輸端點來建立傳輸任務。
4. 在 [建立移轉任務] 頁面上，設定移轉工作的屬性。如果您之前尚未建立任何移轉任務，您可以選擇「建立移轉」任務選項來建立第一個移轉任務。

   - 在此頁面上，輸入轉移任務的基本信息，包括轉移任務名稱，描述和密鑰。

   > **Note**
   >
   > - 使用透過資料傳輸端點定義的KMS金鑰加密密碼。密碼應包含使用和金鑰存取大型主機上資料集所需的大型主機認證。`userId` `password`如需詳細資訊，請參閱 AWSSecrets Manager 密碼。
   > - 您必須使用下列以資源為基礎的原則來設定密鑰，以便 AWS 大型主機現代化服務可以存取它以執行資料傳輸工作。
   >
   > ```
   > {
   >   "Version" : "2012-10-17",
   >   "Statement" : [ {
   >     "Effect" : "Allow",
   >     "Principal" : {
   >         "Service" : "m2.amazonaws.com"
   >     },
   >     "Action" : [ "secretsmanager:GetSecretValue",
   >                  "secretsmanager:DescribeSecret" ],
   >     "Resource" : "*"
   >   } ]
   > }
   > ```

   > **Note**
   >
   > 目前支援傳輸的資料集大小上限為 90 GB。

   - 接下來，選取要從大型主機傳輸目標資料集的目標 Amazon S3 儲存貯體位置。
   - 將選取先前選擇的資料傳輸端點。您也可以從可用端點選取另一個端點。
5. 選擇 Next (下一步)。

6. 在 [新增資料集] 頁面上，在 [搜尋大型主機資料集] 中輸入您的查詢，以搜尋大型主機中要包含在傳輸工作中的資料集。選取檢視資料集。

   下列萬用字元符號可用作大型主機資料集搜尋條件的一部分：

   - 作為限定詞的單一星號 (*) (在期間之間或最後一個週期之後) 與該位置中的單一限定詞相符。
   - 限定詞中的單一星號 (*) 符合該位置中的零個或多個字元。
   - 作為限定詞的雙星號 (**) (在期間之間或最後一個週期之後) 與該位置中的零個或多個限定元相符。
   - 限定詞中的雙星號 (**) 不是有效的查詢。
   - 單一百分比符號 (%) 符合該位置中的任何單一英數字元或國家字元。在每個限定詞中，您最多可以使用百分之八的符號。

   > **Important**
   >
   > 我們建議您一律以句點結束搜尋條件，然後加上雙星號 (.**)，然後視需要進一步調整搜尋範圍。

   如需萬用字元規則的詳細資訊，請參閱IBM文件中的篩選資料集名稱。

7. 這些資料集將載入「大型主機資料集」區段下，您可以在其中搜尋或選擇一或多個要設定字碼頁轉換的資料集。這些選擇的資料集將顯示在 [新增的資料集] 區段中。

   > **Note**
   >
   > 您可以從多個搜尋查詢中選取資料集，並將其新增至傳送任務。

8. 在 [新增的資料集] 區段中，您需要為每個選取的資料集手動選取原始碼頁和目標字碼頁。原始碼頁是來源資料集格式，目標字碼頁是用於轉換資料集並將其存放在目標 Amazon S3 儲存貯體中的目標資料集格式。
9. 確認來源和目標字碼頁之後，請選擇「下一步」。
10. 在「檢閱並建立」頁面上，您可以檢閱或編輯移轉任務的資訊。
11. 然後，選擇創建傳輸任務。

檢視移轉任務

若要在檔案傳輸中檢視傳輸工作，您必須遵循大型主機現代化 AWS 主控台中的下列步驟。

若要檢視移轉工作

1. 在開啟大 AWS 型主機現代化主控台。https://console.aws.amazon.com/m2/
2. 在選擇 AWS 區域 器中，選擇您要將檔案從大型主機傳輸到 Amazon S3 儲存貯體的區域。
3. 在 [傳送工作] 頁面上，選取資料傳輸端點以檢視您的傳輸工作。
4. 對於具有預先存在的移轉工作的端點，這些工作會顯示在「傳送工作」區段下。您可以從此清單中選擇檢視任何移轉任務的詳細資訊。

教學課程：開始使用AWS大型主機現代化檔案傳輸

AWS大型主機現代化檔案傳輸可讓您傳輸和轉換大型主機資料集，以進行大型主機現代化、移轉和擴充使用案例。

請遵循本教學課程中的步驟，瞭解AWS大型主機現代化檔案傳輸的運作方式。

概觀

文件傳輸包括以下內容：

1. 要安裝在來源大型主機上的代理程式。
2. 直接從大型主機現代化管理服務 AWS 主控台存取資料集探索、傳輸和轉換功能。

身為使用者，您可以將資料集從大型主機傳輸到 Amazon S3 儲存貯體。

主題

- 步驟 1：將代理程式二進位檔 tar 套件從傳輸 AWS 到大型主機邏輯分割區
- 步驟 2：在來源大型主機上設定檔案傳輸代理程式
- 步驟 3：建立資料傳輸端點
- 步驟 4：建立轉移任務
- 步驟 5：查看轉移任務進度

步驟 1：將代理程式二進位檔 tar 套件從傳輸 AWS 到大型主機邏輯分割區

從 M2 代理程式 tar 連結下載 tar 檔案。

步驟 2：在來源大型主機上設定檔案傳輸代理程式

在此步驟中，您可以在AWS來源大型主機上設定並啟動大型主機現代化檔案傳輸代理程式。需要代理程式才能促進檔案傳輸服務功能與來源大型主機之間的通訊。每個大型主機至少需要一個代理程式。您可以啟動多個代理程式，以提高可用性和增強的延展性。

依照the section called “安裝檔案傳輸代理程式”指南中的指示完成大型主機上的檔案傳輸代理程式安裝。

步驟 3：建立資料傳輸端點

依照the section called “建立資料傳輸端點”頁面上的步驟建立新的資料傳輸端點。

步驟 4：建立轉移任務

按照the section called “建立移轉任務”頁面上的步驟來創建和管理您的移轉任務。

步驟 5：查看轉移任務進度

您可以在大型主機現代化 AWS 主控台中檢視移轉任務的進度。如需詳細資訊，請參閱the section called “檢視移轉任務”章節。

AWS大型主機現代化檔案傳輸中支援的來源與目標編碼

AWS大型主機現代化檔案傳輸支援各種資料集類型和字碼頁轉換選項。

大型主機資料集類型

AWS大型主機現代化檔案傳輸支援下列大型主機資料集類型：

- 非VSAM：順序 (PS)、PDS、GDS、GDG
- VSAM類型：KSDS

支持的代碼頁

AWS大型主機現代化檔案傳輸支援下列字碼頁以進行資料集轉換 (從/到)：

「BIG5", "BIG5_HKSCS", "CESU_8"," _ EUC 日本 "," EUC _ "", "" ",GB18" "," GB2312 "," GBK "", "01 IBM 144」, "IBM01144」, "IBM01144」, "IBM01143」, "IBM01144」, "," IBM 01144」, "026"、IBM" 047 "、IBM" ""、"IBM"、"" "IBM"、"" "、IBM" 0」IBM IBM1 IBM1 IBM273 IBM277 IBM278 IBM28 IBM284 IBM285 IBM29,「IBM297」,「IBM420」,「IBM424」,「IBM437」, 「IBM500」,「IBM775」,「IBM850」,「IBM852」,「IBM855」,「IBM857」,「」 IBM86,「」, IBM861「」, IBM862「」, IBM863「」, IBM864「」,「IBM865」,「IBM866」,「IBM868」, 「IBM869」,「」 IBM87,「IBM871」,「」, IBM918「」, THAI「」,「IBM_」,「ISO_2022 ISO _CN」,「2」,「,」,「ISO,」ISO ISO ISOISO_8859_15 ",", "", "ISO_8859_ ISO 3",", "" ISO _8859_ ISO 9」, "" "", "" "" "," "ISO_8859_ ISO 7"," "ISO_8859_ ISO 8",", "U」,「ISO_」, 「」,「美國」,「_ 16」,「_ JIS 十六」,「_16」,「_」,「32」,「」,「_ 32」,「JIS_8」, 「KOI8KOI8SHIFTJISTISASCIIUTFUTFUTFUTFUTFUTFUTFWINDOWSWINDOWS_125」, "" "", WINDOWS "WINDOWS_125」,」, "WINDOWS_125」, "" WINDOWS WINDOWS _1256 "," "," "_ _ _ EUC 日本",「X WINDOWS _」,「X WINDOWS _」,「X」,「X WINDOWS _」,「X _」, 「X BIG5 HKSCS _」,「 「,「X BIG5 _ IBM1 046SOLARIS」,「X EUCJP _ 09」,「XOPEN」, 「X」,「XLINUX」,「XEUC」,「X」,「X」,「X」,「X」IBM1 IBM1 IBM1 IBM1 IBM1112 IBM1122 IBM1123 IBM1124 IBM1129,「X」,「XIBM1166」,「XIBM1364」,「X」,「XIBM1381」, 「XIBM1383」,「X_ IBM3 00IBM29626C」,「X」,「X」,「X」,「XIBM33722」,「XIBM737」, 「X_IBM833」,「X_IBM834」,「X」,「XIBM856」,「X」,「XIBM874」,「XIBM875」,「X」 「,「X_IBM921」,「X_IBM922」,「X_」,「X_IBM933」,「X_IBM935」,「X_IBM937」,「X_」, 「X_IBM939」,「X_IBM942」,「X_ IBM93 0IBM942C」,「X_IBM943」IBM943C IBM948 IBM949 IBM949C IBM95IBM964「,「X IBM97 0」,「X」,「XISCII91」,「XISO」,「X」,「XCNS」, 「XISO」,「X」,「X」,「X」,「X」,「XISO」,「X」,「X」,「」 「,「XJISAUTODETECT」,「X」, JOHAB「X」,「XMACARABIC」,「XMACCENTRALEUROPE」,「X」,「XMACCROATIAN」, 「XMACCYRILLIC」,「X」, MACDINGBAT「X」,「X_ JIS 013MACGREEK」,「X _ 0MACHEBREW」,「X _ 0MACICELAND」MACROMAN MACROMANIA MACSYMBOL MACTHAI MACTURKISH MACUKRAINE MS932 MS95 HKSCSMS950 HKSCS _ 經驗值」,「X」MSWIN, 「X」,「XPCK」,「X SJIS _ 02」,「X UTF _ 16」,「BOM」,「X UTF _」,「X _BOM」,「X UTF _BOM」,「_ _ WINDOWS 日」WINDOWS WINDOWS WINDOWS WINDOWS WINDOWS ISO2

AWS 大型主機現代化中的安全性

雲安全 AWS 是最高的優先級。身為 AWS 客戶，您可以從資料中心和網路架構中獲益，該架構專為滿足對安全性最敏感的組織的需求而打造。

安全是 AWS 與您之間共同承擔的責任。共同責任模型將其描述為雲端的安全性和雲端中的安全性：

- 雲端的安全性 — AWS 負責保護在 AWS 雲端中執行 AWS 服務的基礎架構。 AWS 還為您提供可以安全使用的服務。若要了解適用於 AWS 大型主機現代化的法規遵循計劃，請參閱AWS合規計劃合規計劃AWS服務範圍。
- 雲端中的安全性 — 您的責任取決於您使用的 AWS 服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規

本文件可協助您了解如何在使用 AWS 大型主機現代化時套用共同的責任模型。它 AWS 說明如何設定大型主機現代化，以符合您的安全性和合規性目標。您也會學到如何使用其他可協助您監控和保護 AWS 大型主機現代化資源的 AWS 服務。

AWS 大型主機現代化提供自己IAM受保護的資源 (應用程式、環境、部署等)，這些資源是 AWS 大型主機現代化管理資源，政策必須允許任何動作。IAM

AWS 重新平台的大型主機現代化也受到保護。IAMIAM也會透過標準原IAM則，針對已定義資源 (衍生自原始大型主機環境) 的特定動作，授與或拒絕主參與者的權限。當應用程 AWS 式在受保護的資源上嘗試此類動作時，大型主機現代化重新平台執行階段會呼叫IAM授權服務。IAM將會根據標準IAM原則評估機制傳回允許/拒絕。

目錄

- AWS 大型主機現代化中的資料保護
- AWS 大型主機現代化的 Identity and Access Management
- 適用於 AWS 大型主機現代化的合規性驗證
- AWS 大型主機現代化的韌性
- 基礎結構安全 AWS Mainframe Modernization
- 使 AWS Mainframe Modernization 用 AWS PrivateLink 介面端點存取

AWS 大型主機現代化中的資料保護

AWS 共同責任模型適用於大型 AWS 主機現代化中的資料保護。如此模型中所述， AWS 負責保護執行所有 AWS 雲端. 您負責維護在此基礎設施上託管內容的控制權。您也同時負責所使用 AWS 服務 的安全組態和管理任務。如需有關資料隱私權的詳細資訊，請參閱資料隱私權FAQ。 如需歐洲資料保護的相關資訊，請參閱AWS 安全性GDPR部落格上的AWS 共同責任模型和部落格文章。

基於資料保護目的，我們建議您使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 保護 AWS 帳戶 認證並設定個別使用者。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 對每個帳戶使用多重要素驗證 (MFA)。
- 使用SSL/TLS與 AWS 資源溝通。我們需要 TLS 1.2 並推薦 TLS 1.3。
- 使用設定API和使用者活動記錄 AWS CloudTrail。
- 使用 AWS 加密解決方案以及其中的所有默認安全控制 AWS 服務。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Amazon S3 的敏感資料。
- 如果 AWS 透過命令列介面或存取時需要 FIPS 140-3 驗證的密碼編譯模組API，請使用端點。FIPS 如需有關可用FIPS端點的詳細資訊，請參閱聯邦資訊處理標準 (FIPS) 140-3。

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的文字欄位中，例如名稱欄位。這包括當您使用 AWS 主控台、API、或 AWS 服務 使用大型主機現代化或其他工作時。 AWS CLI AWS SDKs您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供URL給外部伺服器，我們強烈建議您不要在中包含認證資訊，URL以驗證您對該伺服器的要求。

AWS 大型主機現代化收集的資料

AWS 大型主機現代化會從您那裡收集數種類型的資料：

- `Application configuration`：這是您建立用來設定應用程式的JSON檔案。它包含您對 AWS 大型主機現代化提供的不同選項的選擇。此檔案也包含相依資 AWS 源的資訊，例如儲存應用程式成品的 Amazon 簡單儲存服務路徑，或儲存資料庫登入資料 AWS Secrets Manager 的 Amazon 資源名稱 (ARN)。
- `Application executable (binary)`：這是您編譯的二進製文件，您打算在 AWS 大型主機現代化上部署。
- `Application JCL or scripts`：此原始程式碼會代表您的應用程式管理批次工作或其他處理。

- `User application data`：匯入資料集時， AWS 大型主機現代化會將它們儲存在關聯式資料庫中，以便您的應用程式可以存取它們。
- `Application source code`：透過 Amazon AppStream 2.0， AWS 大型主機現代化為您提供編寫和編譯程式碼的開發環境。

AWS 大型主機現代化將此資料原生儲存在. AWS AWS 我們向您收集的資料存放在大型主機現代化管理的 Amazon S3 儲存貯體中。部署應用程式時， AWS 大型主機現代化會將資料下載到 Amazon 彈性區塊存放區支援的 Amazon 彈性運算雲端執行個體。觸發清理時，資料會從 Amazon EBS 磁碟區和 Amazon S3 中移除。Amazon EBS 磁碟區是單一租用的，這意味著一個客戶可以使用一個執行個體。永遠不會共用執行個體。刪除執行階段環境時，也會刪除 Amazon EBS 磁碟區。刪除應用程式時，會從 Amazon S3 刪除成品和組態。

應用程式日誌存放在 Amazon 中 CloudWatch。客戶應用程式記錄訊息 CloudWatch 也會匯出至。記 CloudWatch 錄檔可能包含客戶敏感資料，例如商務資料或偵錯訊息中的安全性資訊)。如需詳細資訊，請參閱使用 Amazon 監控 AWS 大型主機現代化 CloudWatch。

此外，如果您選擇將一個或多個 Amazon 彈性檔案系統或 Amazon FSx 檔案系統附加到執行時期環境，這些系統中的資料將存放在其中 AWS。如果您決定停止使用檔案系統，則需要清理這些資料。

當您將資料放置在 AWS 大型主機現代化用於應用程式部署和資料集匯入的 Amazon S3 儲存貯體時，您可以使用所有可用的 Amazon S3 加密選項來保護資料。此外，如果您將一或多個檔案系統附FSx加到執行階段環境，則可以使用 Amazon EFS 和 Amazon 加密選項。

AWS 大型主機現代化服務的靜態資料加密

AWS 大型主機現代化與 AWS Key Management Service 整合，可在永久存放資料的所有相依資源上提供透明的伺服器端加密 (SSE)，也就是 Amazon 簡單儲存服務、Amazon DynamoDB 和 Amazon 彈性區塊存放區。 AWS 大型主機現代化可在中為您建立和管理對稱加密 AWS KMS 金鑰。 AWS KMS

依預設加密靜態資料，有助於降低保護敏感資料所涉及的營運開銷和複雜性。同時，它可讓您遷移需要嚴格加密合規性和法規要求的應用程式。

當您建立執行階段環境和應用程式時，您無法停用此層加密或選取替代的加密類型。

您可以針對 AWS 大型主機現代化應用程式和執行時期環境使用自己的客戶受管金鑰來加密 Amazon S3 和 Amazon 資源。EBS

對於 AWS 大型主機現代化應用程式，您可以使用此金鑰來加密應用程式定義以及其他應用程式資源 (例如JCL檔案)，這些資源會儲存在服務帳戶中建立的 Amazon S3 儲存貯體中。如需詳細資訊，請參閱建立應用程式 。

對於您的 AWS 大型主機現代化執行時期環境， AWS 大型主機現代化會使用客戶受管金鑰來加密建立的 Amazon EBS 磁碟區，並附加到 AWS 大型主機現代化 Amazon EC2 執行個體 (也在服務帳戶中)。如需詳細資訊，請參閱建立執行階段環境。

> **Note**
>
> DynamoDB 資源一律會使用 AWS 大型主機現代化服務帳戶 AWS 受管金鑰 中的加密。您無法使用客戶受管金鑰加密 DynamoDB 資源。

AWS 大型主機現代化使用客戶管理的金鑰執行下列工作：

- 重新部署應用程式。
- 取代 AWS 大型主機現代化 Amazon 執行個體。EC2

AWS 大型主機現代化不會使用客戶受管金鑰來加密 Amazon Relational Database Service 或 Amazon Aurora 資料庫、Amazon 簡單佇列服務佇列和為支援 AWS 大型主機現代化應用程式而建立的 Amazon ElastiCache 快取，因為它們都不包含客戶資料。

如需更多資訊，請參閱 AWS Key Management Service 開發人員指南中的客戶受管金鑰。

下表摘要說明 AWS 大型主機現代化如何加密您的機密資料。

資料類型	AWS 受管金鑰 加密	客戶管理的金鑰加密
`Definition` 包含特定應用程式的定義。	已啟用	已啟用
`EnvironmentSummary` 包含執行階段環境的相關資訊。	已啟用	已啟用
`ApplicationSummary` 包含 AWS 大型主機現代化應用程式的相關資訊。	已啟用	已啟用
`DeploymentSummary`	已啟用	已啟用

資料類型	AWS 受管金鑰 加密	客戶管理的金鑰加密
包含 AWS 大型主機現代化應用程式部署的相關資訊。		

> **Note**
>
> AWS 大型主機現代化可自動啟用靜態加密功能，免費 AWS 受管金鑰 保護您的機密資料。但是，使用客戶管理的金鑰需要 AWS KMS 支付費用。如需定價的詳細資訊，請參閱 AWS Key Management Service 定價。

如需詳細資訊 AWS KMS，請參閱 AWS Key Management Service。

AWS 大型主機現代化如何使用補助金 AWS KMS

AWS 大型主機現代化需要授權才能使用客戶管理的金鑰。

當您建立應用程式或執行階段環境，或在使用客戶管理金鑰加密的 AWS 大型主機現代化中部署應用程式時， AWS 大型主機現代化會將要求傳送至，以代表您建立授權。CreateGrant AWS KMS中的贈款 AWS KMS 是用來授予 AWS 大型主機現代化存取客戶帳戶中KMS金鑰的權限。

AWS 大型主機現代化需要授權，才能將客戶管理的金鑰用於下列內部作業：

- 傳送DescribeKey AWS KMS 要求，以確認建立應用程式、執行階段環境或應用程式部署時所輸入的對稱客戶管理金鑰 ID 是否有效。
- 傳送GenerateDataKey請求，以加密連接 AWS KMS 到託管 AWS 大型主機現代化EC2執行時環境之 Amazon 執行個體的 Amazon EBS 磁碟區。
- 發送解密請求 AWS KMS 以解密 Amazon 上的加密內容EBS。

AWS 大型主機現代化使用 AWS KMS 授權來解密儲存在 Secrets Manager 中的密碼，以及建立執行階段環境、建立或重新部署應用程式，以及建立部署時。 AWS 大型主機現代化建立的授權支援下列作業：

- 創建或更新運行時環境授予：
  - 解密
  - 加密

  - ReEncryptFrom
  - ReEncryptTo
  - GenerateDataKey
  - DescribeKey
  - CreateGrant
- 建立或重新部署應用程式授權：
  - GenerateDataKey
- 建立部署授權：
  - 解密

您可以隨時撤銷授予的存取權，或移除服務對客戶受管金鑰的存取權。如果這樣做， AWS 大型主機現代化將無法存取客戶管理金鑰加密的任何資料，這會影響依賴資料的作業。例如，如果 AWS 大型主機現代化嘗試存取由客戶管理的金鑰加密的應用程式定義，而不授與該金鑰，則應用程式建立作業將會失敗。

AWS 大型主機現代化會收集使用者應用程式組態 (JSON檔案) 和成品 (二進位檔案和可執行檔)。它也會建立中繼資料，以追蹤用於 AWS 大型主機現代化作業的各種實體，並建立記錄檔和指標。客戶可見的日誌和指標包括：

- CloudWatch 反映應用程序和運行時引擎（ AWS 藍光時代或微焦點）的日誌。
- CloudWatch 操作儀表板的指標。

此外， AWS 大型主機現代化還會收集有關服務的計量、活動報告等使用情況資料和指標。這些數據不是客戶可見的。

AWS 大型主機現代化會根據資料類型，將此資料儲存在不同的位置。您上傳的客戶資料會存放在 Amazon S3 儲存貯體中。服務資料同時存放在 Amazon S3 和 DynamoDB 中。部署應用程式時，資料和服務資料都會下載到 Amazon EBS 磁碟區。如果您選擇將 Amazon EFS 或 Amazon FSx 儲存連接到執行階段環境，則存放在這些檔案系統中的資料也會下載到 Amazon EBS 磁碟區。

預設為靜態加密。您無法停用或變更它。目前，您也無法變更其組態。

建立客戶受管金鑰

您可以使用 AWS Management Console 或建立對稱的客戶管理金鑰。 AWS KMS APIs

建立對稱客戶受管金鑰

請依照《AWS Key Management Service 開發人員指南》中建立對稱客戶受管金鑰的步驟進行。

金鑰政策

金鑰政策會控制客戶受管金鑰的存取權限。每個客戶受管金鑰都必須只有一個金鑰政策，其中包含決定誰可以使用金鑰及其使用方式的陳述式。在建立客戶受管金鑰時，可以指定金鑰政策。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的管理客戶受管金鑰的存取。

若要將客戶受管金鑰與 AWS 大型主機現代化資源搭配使用，金鑰原則必須允許下列API作業：

- kms:CreateGrant：新增客戶受管金鑰的授權。授與指定KMS金鑰的控制權存取權，允許存取授與作業 AWS 大型主機現代化需要的權限。如需有關使用授權的詳細資訊，請參閱開AWS Key Management Service 發人員指南。

  這可讓 AWS 大型主機現代化執行下列作業：

  - 呼叫 GenerateDataKey 以產生加密的資料金鑰並加以儲存，因為資料金鑰不會立即用來加密。
  - 呼叫 Decrypt 以使用儲存的加密資料金鑰來存取加密的資料。
  - 設定退休本金以允許服務。RetireGrant
- kms:DescribeKey— 提供客戶管理的金鑰詳細資料，以允許 AWS 大型主機現代化驗證金鑰。

AWS 大型主機現代化需要kms:CreateGrant客戶金鑰原則中的kms:DescribeKey權限。 AWS 大型主機現代化使用此原則為自己建立授權。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "Enable IAM User Permissions",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::AccountId:role/ExampleRole"
        },
        "Action": [
            "kms:CreateGrant",
            "kms:DescribeKey"
        ],
        "Resource": "*"
    }]
}
```

> **Note**
>
> 上述範例`Principal`中顯示的角色是您用於 AWS 大型主機現代化作業 (例如和) 的角色。`CreateApplication` `CreateEnvironment`

如需有關在政策中指定許可的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》。

如需有關故障診斷金鑰存取的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》。

指定 AWS 大型主機現代化的客戶管理金鑰

您可以為下列資源指定客戶管理的金鑰：

- 應用程式
- 環境

建立資源時，您可以輸入 KMSID 來指定金鑰， AWS 大型主機現代化用來加密資源所儲存的機密資料。

- KMSID — 客戶管理金鑰的金鑰識別碼。輸入金鑰 ID、金鑰ARN、別名或別名ARN。

您可以使用 AWS Management Console 或指定客戶管理的金鑰 AWS CLI。

若要在中建立執行階段環境時指定客戶管理的金鑰 AWS Management Console，請參閱建立 AWS 大型主機現代化執行階段環境。若要在中建立應用程式時指定您的客戶管理金鑰 AWS Management Console，請參閱建立 AWS Mainframe Modernization 應用程式。

若要在使用建立執行階段環境時新增客戶管理的金鑰 AWS CLI，請指定`kms-key-id`參數，如下所示：

```
aws m2 create-environment –engine-type microfocus –instance-type M2.m5.large
--publicly-accessible –engine-version 7.0.3 –name test
--high-availability-config desiredCapacity=2
--kms-key-id myEnvironmentKey
```

若要在使用建立應用程式時新增客戶管理的金鑰 AWS CLI，請指定`kms-key-id`參數，如下所示：

```
aws m2 create-application –name test-application –description my description
--engine-type microfocus
```

```
--definition content="$(jq -c . raw-template.json | jq -R)"
--kms-key-id myApplicationKey
```

AWS 大型主機現代化加密內容

加密內容是一組選用的金鑰值對，包含資料的其他相關內容資訊。

AWS KMS 使用加密內容作為其他驗證資料，以支援已驗證的加密。當您在加密資料的要求中包含加密內容時，會將加密內容 AWS KMS 繫結至加密的資料。若要解密資料，您必須在請求中包含相同的加密內容。

AWS 大型主機現代化加密內容

AWS 大型主機現代化在與應用程式相關的所有 AWS KMS 密碼編譯作業中使用相同的加密內容 (建立應用程式並建立部署)，其中金鑰所在，aws:m2:app而且值是應用程式的唯一識別碼。

Example

```
"encryptionContextSubset": {
        "aws:m2:app": "a1bc2defabc3defabc4defabcd"
}
```

使用加密內容進行監控

當您使用對稱的客戶管理金鑰來加密應用程式或執行階段環境時，您也可以在稽核記錄和記錄中使用加密內容，以識別客戶管理金鑰的使用方式。

使用加密內容控制對客戶受管金鑰的存取

您可以在金鑰政策和IAM政策中使用加密內容，conditions以控制對稱客戶管理金鑰的存取。您也可以在授予中使用加密內容條件。

AWS 大型主機現代化使用授權中的加密內容限制，以控制對您帳戶或區域中客戶管理金鑰的存取。授予條件會要求授予允許的操作使用指定的加密內容。下列範例是 AWS 大型主機現代化在建立應用程式時利用來加密應用程式成品的授權。

```
//This grant is retired immediately after create application finish
{
   "grantee-principal": m2.us-west-2.amazonaws.com,
   "retiring-principal": m2.us-west-2.amazonaws.com,
   "operations": [
       "GenerateDataKey"
```

```
    ]
    "condition": {
         "encryptionContextSubset": {
             “aws:m2:app”: “a1bc2defabc3defabc4defabcd”
    }
}
```

監控 AWS 大型主機現代化的加密金鑰

當您將 AWS KMS 客戶受管金鑰與 AWS 大型主機現代化資源搭配使用時，您可以使用AWS CloudTrail或 Amazon CloudWatch Logs 追蹤 AWS 大型主機現代化傳送到的請求。 AWS KMS

執行階段環境的範例

下列範例是 AWS 大型主機現代化所呼叫的KMS作業 `DescribeKey` `CreateGrantGenerateDataKey`，`Decrypt`以存取由客戶管理的金鑰加密之資料的 AWS CloudTrail 事件、、和監視：

DescribeKey

AWS 大型主機現代化使用`DescribeKey`作業來驗證帳 AWS KMS 戶和區域中是否存在與您執行階段環境相關聯的客戶管理金鑰。

下面的範例事件會記錄 `DescribeKey` 操作：

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
```

```
                "creationDate": "2022-12-06T19:40:26Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-12-06T20:23:43Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "205.251.233.182",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "keyId": "00dd0db0-0000-0000-ac00-b0c000SAMPLE"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_256_GCM_SHA384",
        "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
    },
    "sessionCredentialFromConsole": "true"
}
```


CreateGrant

當您使用 AWS KMS 客戶管理的金鑰來加密執行階段環境時， AWS 大型主機現代化會代表您傳送數個CreateGrant要求，以執行必要的作業。KMS AWS 大型主機現代化建立的部分授權會在使用後立即淘汰。當您刪除執行階段環境時，其他人則會淘汰。

下列範例事件會記錄與建立環境工CreateGrant作流程相關聯之 Lambda 執行角色的作業。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T20:11:45Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "m2.us-west-2.amazonaws.com"
    },
    "eventTime": "2022-12-06T20:23:09Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "m2.us-west-2.amazonaws.com",
    "userAgent": "m2.us-west-2.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "operations": [
            "Encrypt",
            "Decrypt",
            "ReEncryptFrom",
            "ReEncryptTo",
            "GenerateDataKey",
            "GenerateDataKey",
            "DescribeKey",
            "CreateGrant"
```

```
        ],
        "granteePrincipal": "m2.us-west-2.amazonaws.com",
        "retiringPrincipal": "m2.us-west-2.amazonaws.com"
    },
    "responseElements": {
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

下列範例事件會記錄 Auto Scaling 群組服務連結角色的`CreateGrant`作業。與「建立環境」工作流程相關聯的 Lambda 執行角色會呼叫此`CreateGrant`作業。它會授與執行角色的權限，以針對 Auto Scaling 群組的服務連結角色建立子授權。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA3YPCLM65MZFUPM4JO:EnvironmentWorkflow-alpha-
CreateEnvironmentLambda7-HfxDj5zz86tr",
        "arn": "arn:aws:sts::111122223333:assumed-role/EnvironmentWorkflow-
alpha-CreateEnvironmentLambdaS-1AU4A8VNQEEKN/EnvironmentWorkflow-alpha-
CreateEnvironmentLambda7-HfxDj5zz86tr",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
```

```
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:iam::111122223333:role/EnvironmentWorkflow-alpha-
CreateEnvironmentLambdaS-1AU4A8VNQEEKN",
                "accountId": "111122223333",
                "userName": "EnvironmentWorkflow-alpha-
CreateEnvironmentLambdaS-1AU4A8VNQEEKN"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T20:22:28Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-12-06T20:23:09Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "54.148.236.160",
    "userAgent": "aws-sdk-java/2.18.21 Linux/4.14.255-276-224.499.amzn2.x86_64
 OpenJDK_64-Bit_Server_VM/11.0.14.1+10-LTS Java/11.0.14.1 vendor/Amazon.com_Inc. md/
internal exec-env/AWS_Lambda_java11 io/sync http/Apache cfg/retry-mode/legacy",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "operations": [
            "Encrypt",
            "Decrypt",
            "ReEncryptFrom",
            "ReEncryptTo",
            "GenerateDataKey",
            "GenerateDataKey",
            "DescribeKey",
            "CreateGrant"
        ],
        "granteePrincipal": "m2.us-west-2.amazonaws.com",
        "retiringPrincipal": "m2.us-west-2.amazonaws.com"
    },
    "responseElements": {
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE",
```

```
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_256_GCM_SHA384",
        "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
    }
}
}
```

GenerateDataKey

當您為執行時期環境資源啟用 AWS KMS 客戶受管金鑰時，Auto Scaling 會建立一個唯一金鑰，用於加密與執行階段環境關聯的 Amazon EBS 磁碟區。它會將要`GenerateDataKey`求傳送至 AWS KMS 指定資源的 AWS KMS 客戶管理金鑰。

下面的範例事件會記錄 `GenerateDataKey` 操作：

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA3YPCLM65EEXVIEH7D:AutoScaling",
        "arn": "arn:aws:sts::111122223333:assumed-role/AWSServiceRoleForAutoScaling/
AutoScaling",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
```

```
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:iam::111122223333:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling",
                "accountId": "111122223333",
                "userName": "AWSServiceRoleForAutoScaling"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T20:23:16Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "autoscaling.amazonaws.com"
    },
    "eventTime": "2022-12-06T20:23:18Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "autoscaling.amazonaws.com",
    "userAgent": "autoscaling.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "aws:ebs:id": "vol-080f7a32d290807f3"
        },
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "numberOfBytes": 64
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
```

```
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

Decrypt

當您存取加密的執行階段環境時，Amazon 會EBS呼叫Decrypt作業以使用儲存的加密資料金鑰存取加密的資料。

下面的範例事件會記錄 Decrypt 操作：

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "ebs.amazonaws.com"
    },
    "eventTime": "2022-12-06T20:23:22Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "ebs.amazonaws.com",
    "userAgent": "ebs.amazonaws.com",
    "requestParameters": {
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "encryptionContext": {
            "aws:ebs:id": "vol-080f7a32d290807f3"
        }
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
```

```
    "recipientAccountId": "111122223333",
    "sharedEventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventCategory": "Management"
}
```

應用範例

下列範例是 AWS 大型主機現代化所呼叫GenerateDataKey的KMS作業，以存取由客戶管理的金鑰加密之資料的 AWS CloudTrail 事件，以CreateGrant及監控作業：

CreateGrant

當您使用 AWS KMS 客戶受管金鑰來加密應用程式資源時，Lambda 執行角色會代表您傳送CreateGrant請求，以存取您 AWS 帳戶中的KMS金鑰。此授權可讓 Lambda 執行角色使用客戶受管金鑰將客戶應用程式資源上傳到 Amazon S3。此授權會在建立應用程式之後立即淘汰。

下面的範例事件會記錄 CreateGrant 操作：

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T21:51:45Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "m2.us-west-2.amazonaws.com"
    },
```

```
    "eventTime": "2022-12-06T22:47:04Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "m2.us-west-2.amazonaws.com",
    "userAgent": "m2.us-west-2.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "constraints": {
            "encryptionContextSubset": {
                "aws:m2:app": "a1bc2defabc3defabc4defabcd"
            }
        },
        "retiringPrincipal": "m2.us-west-2.amazonaws.com",
        "operations": [
            "GenerateDataKey"
        ],
        "granteePrincipal": "m2.us-west-2.amazonaws.com"
    },
    "responseElements": {
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

GenerateDataKey

當您為應用程式資源啟用 AWS KMS 客戶受管金鑰時，Lambda 執行角色會建立金鑰，用來加密客戶資料，並將其上傳到 Amazon 簡單儲存服務。Lambda 執行角色會傳送GenerateDataKey請求給 AWS KMS ，以指定資源的 AWS KMS 客戶受管金鑰。

下面的範例事件會記錄 GenerateDataKey 操作：


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA3YPCLM65CLCEKKC7Z:ApplicationWorkflow-alpha-
CreateApplicationVersion-CstWZUn5R4u6",
        "arn": "arn:aws:sts::111122223333:assumed-role/ApplicationWorkflow-
alpha-CreateApplicationVersion-1IZRBZYDG20B/ApplicationWorkflow-alpha-
CreateApplicationVersion-CstWZUn5R4u6",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:iam::111122223333:role/ApplicationWorkflow-alpha-
CreateApplicationVersion-1IZRBZYDG20B",
                "accountId": "111122223333",
                "userName": "ApplicationWorkflow-alpha-
CreateApplicationVersion-1IZRBZYDG20B"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T23:28:32Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "m2.us-west-2.amazonaws.com"
    },
    "eventTime": "2022-12-06T23:29:08Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "m2.us-west-2.amazonaws.com",
    "userAgent": "m2.us-west-2.amazonaws.com",
```

```
    "requestParameters": {
        "encryptionContext": {
            "aws:m2:app": "a1bc2defabc3defabc4defabcd",
            "aws:s3:arn": "arn:aws:s3:::supernova-processedtemplate-111122223333-us-
west-2/111122223333/a1bc2defabc3defabc4defabcd/1/cics-transaction/ZBNKE35.so"
        },
        "keySpec": "AES_256",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

部署範例

下列範例是 AWS 大型主機現代化所呼叫Decrypt的KMS作業，以存取由客戶管理的金鑰加密之資料的 AWS CloudTrail 事件，以CreateGrant及監控作業：

CreateGrant

當您使用 AWS KMS 客戶受管金鑰來加密部署資源時， AWS 大型主機現代化會代表您傳送兩個CreateGrant要求。第一個授權是針對目前要呼叫的 Lambda 執行角色 ListBatchJobScriptFiles，並在部署完成後立即淘汰。第二次授予是針對 Amazon EC2 範圍縮減執行個體角色，以便 Amazon EC2 可以從 Amazon S3 下載客戶應用程式資源。從執行階段環境中刪除應用程式時，此授權會被淘汰。

下面的範例事件會記錄 CreateGrant 操作：

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T21:51:45Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "m2.us-west-2.amazonaws.com"
    },
    "eventTime": "2022-12-06T23:40:07Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "m2.us-west-2.amazonaws.com",
    "userAgent": "m2.us-west-2.amazonaws.com",
    "requestParameters": {
        "operations": [
            "Decrypt"
        ],
        "constraints": {
            "encryptionContextSubset": {
                "aws:m2:app": "a1bc2defabc3defabc4defabcd"
            }
        },
        "granteePrincipal": "m2.us-west-2.amazonaws.com",
        "retiringPrincipal": "m2.us-west-2.amazonaws.com",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
```

```
    },
    "responseElements": {
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

Decrypt

當您存取部署時，Amazon 會EC2呼叫Decrypt作業，使用存放的加密資料金鑰從 Amazon S3 解密和下載加密的客戶資料。

下面的範例事件會記錄 Decrypt 操作：

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA3YPCLM65BSPZ37E6G:m2-hm-bqe367dxtfcpdbzmnhfzranisu",
        "arn": "arn:aws:sts::111122223333:assumed-role/
SupernovaEnvironmentInstanceScopeDownRole/m2-hm-bqe367dxtfcpdbzmnhfzranisu",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
```

```
                "arn": "arn:aws:iam::111122223333:role/
SupernovaEnvironmentInstanceScopeDownRole",
                "accountId": "111122223333",
                "userName": "SupernovaEnvironmentInstanceScopeDownRole"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T23:19:29Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "m2.us-west-2.amazonaws.com"
    },
    "eventTime": "2022-12-06T23:40:15Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "m2.us-west-2.amazonaws.com",
    "userAgent": "m2.us-west-2.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "aws:m2:app": "a1bc2defabc3defabc4defabcdm",
            "aws:s3:arn": "arn:aws:s3:::supernova-processedtemplate-111122223333-us-
west-2/111122223333/a1bc2defabc3defabc4defabcdm/1/cics-transaction/BBANK40P.so"
        },
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
```

```
}
```

進一步了解

下列資源會提供有關靜態資料加密的詳細資訊。

- 如需 AWS Key Management Service 基本概念的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》。
- 如需有關 AWS Key Management Service的安全最佳實務的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》。

傳輸中加密

對於屬於交易工作負載一部分的互動式應用程式，終端機模擬器與 TN327 0 通訊協定的 AWS 大型主機現代化服務端點之間的資料交換不會在傳輸過程中加密。如果應用程式在傳輸過程中需要加密，您可能需要實作一些額外的通道機制。

AWS 大型主機現代化用HTTPS來加密服務。APIs AWS 大型主機現代化中的所有其他通訊都受到服務VPC或安全性群組的保護，以及。HTTPS AWS 大型主機現代化可傳輸應用程式構件、組態和應用程式資料。應用程式人工因素會從您擁有的 Amazon S3 儲存貯體複製，應用程式資料也是如此。您可以使用 Amazon S3 的連結或在本機上傳檔案來提供應用程式組態。

傳輸中的基本加密預設會設定，但不適用於 TN327 0 通訊協定。 AWS 大型主機現代化用HTTPS於API端點，預設也會設定這些端點。

AWS 大型主機現代化的 Identity and Access Management

AWS Identity and Access Management (IAM) 可協助系統管理員安全地控制 AWS 資源存取權。 AWS 服務 IAM系統管理員控制誰可以驗證 (登入) 和授權 (具有權限) 使用 AWS 大型主機現代化資源。IAM 是一種您 AWS 服務 可以使用，無需額外費用。

主題

- 物件
- 使用身分驗證
- 使用政策管理存取權

- AWS 大型主機現代化如何搭配運作 IAM
- 大型主機現代化的身分識別原則範例 AWS
- 疑難排解 AWS 大型主機現代化身分識別與存取
- 使用 AWS Mainframe Modernization的服務連結角色

物件

根據您在 AWS 大型主機現代化中所做的工作而有所不同 AWS Identity and Access Management (IAM) 的使用方式。

服務使用者 — 如果您使用 AWS 大型主機現代化服務來完成工作，則管理員會為您提供所需的認證和權限。當您使用更多 AWS 大型主機現代化功能來完成工作時，您可能需要額外的權限。了解存取許可的管理方式可協助您向管理員請求正確的許可。如果您無法存取 AWS 大型主機現代化中的功能，請參閱。疑難排解 AWS 大型主機現代化身分識別與存取

服務管理員 — 如果您負責公司的 AWS 大型主機現代化資源，您可能擁有大型主機現代化的完整存取 AWS 權。判斷服務使用者應該存取哪些 AWS 大型主機現代化功能和資源是您的工作。然後，您必須向IAM管理員提交請求，才能變更服務使用者的權限。檢閱此頁面上的資訊，以瞭解的基本概念IAM。若要深入瞭解貴公司如何IAM搭配 AWS 大型主機現代化使用，請參閱。AWS 大型主機現代化如何搭配運作 IAM

IAM系統管理員 — 如果您是IAM系統管理員，您可能想要瞭解如何撰寫原則以管理 AWS 大型主機現代化存取權限的詳細資訊。若要檢視您可以在中使用的 AWS 大型主機現代化身分型原則範例，請參閱。IAM 大型主機現代化的身分識別原則範例 AWS

使用身分驗證

驗證是您 AWS 使用身分認證登入的方式。您必須以IAM使用者身分或假設IAM角色來驗證 (登入 AWS)。 AWS 帳戶根使用者

您可以使用透過 AWS 身分識別來源提供的認證，以聯合身分識別身分登入。 AWS IAM Identity Center (IAM身分識別中心) 使用者、貴公司的單一登入驗證，以及您的 Google 或 Facebook 認證都是聯合身分識別的範例。當您以同盟身分登入時，您的管理員先前會使用IAM角色設定聯合身分識別。當您使 AWS 用同盟存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需有關登入的詳細資訊 AWS，請參閱《AWS 登入 使用指南》 AWS 帳戶中的如何登入您的。

如果您 AWS 以程式設計方式存取，請 AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以使用您的認證以密碼編譯方式簽署您的要求。如果您不使用 AWS 工具，則必須自行簽署要求。如需使用建議的方法自行簽署要求的詳細資訊，請參閱使用IAM者指南中的簽署 AWS API要求。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如， AWS 建議您使用多重要素驗證 (MFA) 來增加帳戶的安全性。若要深入瞭解，請參閱使用AWS IAM Identity Center 者指南中的多重要素驗證和使用多重要素驗證 (MFA) AWS的使用IAM者指南。

AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取該帳戶中的所有資源 AWS 服務和資源。此身分稱為 AWS 帳戶 root 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需需要您以 root 使用者身分登入的完整工作清單，請參閱《使用指南》中的〈需要 root 使用者認證的IAM工作〉。

聯合身分

最佳作法是要求人類使用者 (包括需要系統管理員存取權的使用者) 使用與身分識別提供者的同盟，才能使用臨時認證 AWS 服務 來存取。

聯合身分識別是來自企業使用者目錄的使用者、Web 身分識別提供者、Identi ty Center 目錄，或使用透過身分識別來源提供的認證進行存取 AWS 服務 的任何使用者。 AWS Directory Service同盟身分存取時 AWS 帳戶，他們會假設角色，而角色則提供臨時認證。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，也可以連線並同步至您自己身分識別來源中的一組使用者和群組，以便在所有應用程式 AWS 帳戶 和應用程式中使用。如需IAM身分識別中心的相關資訊，請參閱IAM識別中心是什麼？在《AWS IAM Identity Center 使用者指南》中。

IAM 使用者和群組

IAM使用者是您內部的身分，具 AWS 帳戶 有單一人員或應用程式的特定權限。在可能的情況下，我們建議您仰賴臨時登入資料，而不要建立具有長期認證 (例如密碼和存取金鑰) 的IAM使用者。不過，如果您的特定使用案例需要使用IAM者的長期認證，建議您輪換存取金鑰。如需詳細資訊，請參閱《使用指南》中的「IAM定期輪換存取金鑰」以瞭解需要長期認證的使用案例。

IAM群組是指定IAM使用者集合的身分識別。您無法以群組身分簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有一個名為的群組，IAMAdmins並授與該群組管理IAM資源的權限。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供暫時憑證。要了解更多信息，請參閱《IAM用戶指南》中的創建用戶（而不是角色）的IAM時間。

IAM角色

IAM角色是您 AWS 帳戶 中具有特定權限的身份。它類似於用IAM戶，但不與特定人員相關聯。您可以 AWS Management Console 透過切換角色來暫時擔任中的角色。IAM您可以呼叫 AWS CLI 或 AWS API作業或使用自訂來擔任角色URL。如需有關使用角色方法的詳細資訊，請參閱《使用指南》中的IAM〈使用IAM角色〉。

IAM具有臨時認證的角色在下列情況下很有用：

- 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需聯合角色的相關資訊，請參閱《使用指南》中的〈建立第三方身分識別提供IAM者的角色〉。如果您使用IAM身分識別中心，則需要設定權限集。為了控制身分驗證後可以存取的內IAM容，IAMIdentity Center 會將權限集與中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的許可集。
- 暫時IAM使用者權限 — IAM 使用者或角色可以假定某個IAM角色，暫時取得特定工作的不同權限。
- 跨帳戶存取 — 您可以使用IAM角色允許不同帳戶中的某個人 (受信任的主體) 存取您帳戶中的資源。角色是授予跨帳戶存取權的主要方式。但是，對於某些策略 AWS 服務，您可以將策略直接附加到資源（而不是使用角色作為代理）。若要瞭解跨帳戶存取角色與以資源為基礎的政策之間的差異，請參閱《IAM使用指南》IAM中的〈跨帳號資源存取〉。
- 跨服務訪問 — 有些 AWS 服務 使用其他 AWS 服務功能。例如，當您在服務中撥打電話時，該服務通常會在 Amazon 中執行應用程式EC2或將物件存放在 Amazon S3 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉寄存取工作階段 (FAS) — 當您使用使用IAM者或角色執行中的動作時 AWS，您會被視為主參與者。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS會使用主參與者呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。FAS只有當服務收到需要與其他 AWS 服務 資源互動才能完成的請求時，才會發出請求。在此情況下，您必須具有執行這兩個動作的許可。有關提出FAS請求時的策略詳細信息，請參閱轉發訪問會話。
  - 服務角色 — 服務角色是指服務代表您執行動作所代表的IAM角色。IAM管理員可以從中建立、修改和刪除服務角色IAM。如需詳細資訊，請參閱《IAM使用指南》 AWS 服務中的建立角色以將權限委派給

- 服務連結角色 — 服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM管理員可以檢視 (但無法編輯服務連結角色) 的權限。
- 在 Amazon 上執行的應用程式 EC2 — 您可以使用IAM角色來管理在執行個體上EC2執行的應用程式以及發出 AWS CLI 或 AWS API請求的臨時登入資料。這比在EC2執行個體中儲存存取金鑰更可取。若要將 AWS 角色指派給EC2執行個體並讓其所有應用程式都能使用，請建立附加至執行個體的執行個體設定檔。執行個體設定檔包含角色，可讓執行個體上EC2執行的程式取得臨時登入資料。如需詳細資訊，請參閱使用者指南中的使用IAM角色將許可授與在 Amazon EC2 執行個體上執行的應IAM用程式。

要了解是否使用IAM角色還是用IAM戶，請參閱《用戶指南》中的「IAM創建IAM角色的時機（而不是用戶）」。

使用政策管理存取權

您可以透 AWS 過建立原則並將其附加至 AWS 身分識別或資源來控制中的存取。原則是一個物件 AWS ，當與身分識別或資源相關聯時，會定義其權限。 AWS 當主參與者 (使用者、root 使用者或角色工作階段) 提出要求時，評估這些原則。政策中的許可決定是否允許或拒絕請求。大多數原則會 AWS 以JSON文件的形式儲存在中。如需有關JSON原則文件結構和內容的詳細資訊，請參閱《IAM使用指南》中的策略概觀。JSON

管理員可以使用 AWS JSON策略來指定誰可以存取什麼內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授與使用者對所需資源執行動作的權限，IAM管理員可以建立IAM策略。然後，系統管理員可以將IAM原則新增至角色，使用者可以擔任這些角色。

IAM原則會定義動作的權限，不論您用來執行作業的方法為何。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該原則的使用者可以從 AWS Management Console AWS CLI、或取得角色資訊 AWS API。

身分型政策

以身分識別為基礎的原則是您可以附加至身分識別 (例如使用者、使用IAM者群組或角色) 的JSON權限原則文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要瞭解如何建立以身分識別為基礎的策略，請參閱《IAM使用指南》中的〈建立IAM策略〉。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管理的策略是獨立策略，您可以將其附加到您的 AWS 帳戶. 受管政策包括 AWS 受管政策和客戶管理

的策略。若要了解如何在受管策略或內嵌策略之間進行選擇，請參閱《IAM使用手冊》中的「在受管策略和內嵌策略之間進行選擇」。

資源型政策

以資源為基礎的JSON策略是您附加至資源的政策文件。以資源為基礎的政策範例包括IAM角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

資源型政策是位於該服務中的內嵌政策。您無法在以資源為基礎的策略IAM中使用 AWS 受管政策。

存取控制清單 (ACLs)

存取控制清單 (ACLs) 控制哪些主參與者 (帳戶成員、使用者或角色) 具有存取資源的權限。ACLs類似於以資源為基礎的策略，雖然它們不使用JSON政策文件格式。

Amazon S3 和 Amazon VPC 是支持服務的示例ACLs。 AWS WAF若要進一步了解ACLs，請參閱 Amazon 簡單儲存服務開發人員指南中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他較不常見的原則類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 權限界限 — 權限界限是一項進階功能，您可以在其中設定以身分識別為基礎的原則可授與給IAM實體 (IAM使用者或角色) 的最大權限。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需有關權限界限的詳細資訊，請參閱《IAM使用指南》中的IAM實體的權限界限。
- 服務控制策略 (SCPs) — SCPs 是指定中組織或組織單位 (OU) 最大權限的JSON策略 AWS Organizations。 AWS Organizations 是一種用於分組和集中管理您企業擁 AWS 帳戶 有的多個服務。如果您啟用組織中的所有功能，則可以將服務控制策略 (SCPs) 套用至您的任何或所有帳戶。SCP限制成員帳戶中實體的權限，包括每個帳戶 AWS 帳戶根使用者。如需有關 Organizations 的詳細資訊SCPs，請參閱AWS Organizations 使用指南中的服務控制原則。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱《IAM使用指南》中的工作階段原則。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。若要瞭解如何在涉及多個原則類型時 AWS 決定是否允許要求，請參閱IAM使用指南中的原則評估邏輯。

AWS 大型主機現代化如何搭配運作 IAM

在您用IAM來管理 AWS 大型主機現代化的存取權限之前，請先了解可與大型主機現代化搭配 AWS 使用的IAM功能。

IAM可搭配 AWS 大型主機現代化使用的功能

IAM特徵	AWS 大型主機現代化支援
身分型政策	是
資源型政策	否
政策動作	是
政策資源	是
政策條件索引鍵	是
ACLs	否
ABAC(策略中的標籤)	是
暫時性憑證	是
轉寄存取工作階段 (FAS)	是
服務角色	是
服務連結角色	是

若要深入瞭解大 AWS 型主機現代化和其他 AWS 服務如何搭配大部分IAM功能運作，請參閱使用者指南IAM中的適用AWS 服務。IAM

大型主機現代化的身分識別原則 AWS

支援身分型政策：是

以身分識別為基礎的原則是您可以附加至身分識別 (例如使用者、使用IAM者群組或角色) 的JSON權限原則文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要瞭解如何建立以身分識別為基礎的策略，請參閱《IAM使用指南》中的〈建立IAM策略〉。

使用以IAM身分識別為基礎的策略，您可以指定允許或拒絕的動作和資源，以及允許或拒絕動作的條件。您無法在身分型政策中指定主體，因為這會套用至連接的使用者或角色。若要瞭解可在JSON策略中使用的所有元素，請參閱《使用IAM者指南》中的IAMJSON策略元素參考資料。

大型主機現代化的身分識別原則範例 AWS

若要檢視大型主機現代化身分 AWS 型原則的範例，請參閱。大型主機現代化的身分識別原則範例 AWS

AWS 大型主機現代化中以資源為基礎的政策

支援資源型政策：否

以資源為基礎的JSON策略是您附加至資源的政策文件。以資源為基礎的政策範例包括IAM角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

若要啟用跨帳戶存取，您可以在以資源為基礎的策略中指定一個或多個帳戶中的一個或多個帳戶中的IAM實體作為主體。新增跨帳戶主體至資源型政策，只是建立信任關係的一半。當主參與者和資源不同時 AWS 帳戶，受信任帳戶中的IAM管理員也必須授與主參與者實體 (使用者或角色) 權限，才能存取資源。其透過將身分型政策連接到實體來授與許可。不過，如果資源型政策會為相同帳戶中的主體授予存取，這時就不需要額外的身分型政策。如需詳細資訊，請參閱《IAM使用指南》IAM中的〈跨帳號資源存取〉。

AWS 大型主機現代化的政策動作

支援政策動作：是

管理員可以使用 AWS JSON策略來指定誰可以存取什麼內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

JSON策略Action元素描述了您可以用來允許或拒絕策略中存取的動作。策略動作通常與關聯 AWS API操作具有相同的名稱。有一些例外情況，例如沒有匹配API操作的僅限權限的操作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

若要查看 AWS 大型主機現代化動作清單，請參閱服務授權參考資料中的AWS 大型主機現代化定義的動作。

AWS 大型主機現代化中的原則動作會在動作前使用下列前置詞：

```
m2
```

如需在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [
      "m2:StartApplication",
      "m2:StopApplication"
         ]
```

您也可以使用萬用字元 (*) 來指定多個動作。例如，若要指定開頭是 List 文字的所有動作，請包含以下動作：

```
"Action": "m2:List*"
```

若要檢視大型主機現代化身分 AWS 型原則的範例，請參閱。大型主機現代化的身分識別原則範例 AWS

AWS 大型主機現代化的政策資源

支援政策資源：是

管理員可以使用 AWS JSON策略來指定誰可以存取什麼內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

ResourceJSON原則元素會指定要套用動作的一個或多個物件。陳述式必須包含 Resource 或 NotResource 元素。最佳做法是使用其 Amazon 資源名稱 (ARN) 指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"
```

您可以使用特定 AWS 大型主機現代化資源來識別適用IAM政策的資源，以限制對特ARNs定大型主機現代化資源的存取。如需有關的格式的詳 Amazon 資訊 ARNs，請參閱 AWS 一般參考. ARNs

例如， AWS 大型主機現代化環境具有下列項目。ARN

```
"Resource": "arn:aws:m2:regionId:accountId:env/service-generated-unique-identifier"
```

AWS 大型主機現代化應用程式具有下列項目。ARN

```
"Resource": "arn:aws:m2:regionId:accountId:app/service-generated-unique-identifier"
```

並非所有 AWS 大型主機現代化動作都支援資源層級權限。對於不支援資源層級權限的動作，您必須使用萬用字元 (*)。

下列 AWS 大型主機現代化動作不支援資源層級權限。

```
ListApplications
            ListApplicationVersions
            ListBatchJobDefinitions
            ListBatchJobExecutions
            ListDataSetImportHistory
            ListDataSets
            ListDeployments
            ListEngineVersions
            ListEnvironments
            ListTagsForResource
```

若要查看 AWS 大型主機現代化資源類型及其清單ARNs，請參閱服務授權參考資料中的大型 AWS 主機現代化定義的資源。若要瞭解您可以針AWS 對每個資源指定哪些動作，請參閱大型主機現代化所定義ARN的動作。

若要檢視大型主機現代化身分 AWS 型原則的範例，請參閱。大型主機現代化的身分識別原則範例 AWS

AWS 大型主機現代化API權限：動作、資源和條件參考

當您撰寫可以附加至身分識別 (以IAM身分識別為基礎的原則) 的權限原則時，您可以使用下表作為參考。該表包括以下內容：

- 每個 AWS 大型主機現代化API作業。
- 您可以授與執行動作之權限的對應動作。
- 您可以授與權限的 AWS 資源。

您要在政策的 `Action` 欄位中指定動作，並在政策的 `Resource` 欄位中指定資源值。

您可以在 AWS 大型主機現代化政策中使用 AWS 全域條件金鑰來表示條件。如需索引 AWS 鍵的完整清單，請參閱《IAM使用指南》中的可用全域條件金鑰。

> **Note**
>
> 若要指定動作，請使用`m2:`前置詞，後面接著API作業名稱 (例如，`m2:CreateApplication`)。

AWS 大型主機現代化API和動作所需的權限

AWS 大型主機現代化作業 API	必要權限 (API動作)	資源
CancelBatchJobExecution		應用程式

CreateApplication	`iam:PassRole`
	`kms:DescribeKey`
	`kms:CreateGrant`
	`s3:GetObject`
	`s3:ListBucket`
		應用程式

CreateDataSetImportTask	`m2:CreateDataSetImportTask` `s3:GetObject`	應用程式
CreateDeployment	`elasticloadbalancing:AddTags`	應用程式

AWS 大型主機現代化作業 API	必要權限 (API動作)	資源
	elasticloadbalancing:CreateListener elasticloadbalancing:CreateTargetGroup elasticloadbalancing:RegisterTargets	

AWS 大型主機現代化作業 API	必要權限 (API動作)	資源
CreateEnvironment	ec2:CreateNetworkInterface ec2:CreateNetworkInterfacePermission ec2:DescribeNetworkInterfaces ec2:DescribeSecurityGroups ec2:DescribeSubnets ec2:DescribeVpcAttribute ec2:DescribeVpcs ec2:ModifyNetworkInterfaceAttribute elasticfilesystem:DescribeMountTargets elasticloadbalancing:AddTags elasticloadbalancing:CreateLoadBalancer elasticloadbalancing:DeleteLoadBalancer kms:DescribeKey kms:CreateGrant fsx:DescribeFileSystems	Environment (環境)

AWS 大型主機現代化作業 API	必要權限 (API動作)	資源
	`iam:CreateServiceLinkedRole`	

DeleteApplication	`elasticloadbalancing:DeleteListener`
	`elasticloadbalancing:DeleteTargetGroup`
	`logs:DeleteLogDelivery`
		應用程式

DeleteApplicationFromEnvironment	`elasticloadbalancing:DeleteListener` `elasticloadbalancing:DeleteTargetGroup`	應用程式 環境
DeleteEnvironment	`elasticloadbalancing:DeleteLoadBalancer`	Environment (環境)
GetApplication		應用程式
GetApplicationVersion		應用程式
GetBatchJobExecution		應用程式
GetDataSetDetails		應用程式
GetDataSetImportTask		應用程式
GetDeployment		應用程式
GetEnvironment		Environment (環境)
ListApplications		*

AWS 大型主機現代化作業 API	必要權限 (API動作)	資源
ListApplicationVersions		*
ListBatchJobDefinitions		*
ListBatchJobExecutions		*
ListDataSetImportHistory		*
ListDataSets		*
ListDeployments		*
ListEngineVersions		*
ListEnvironments		*
ListTagsForResource		*
StartApplication		應用程式
StartBatchJob		應用程式
StopApplication		應用程式
TagResource		*
UntagResource		*
UpdateApplication	s3:GetObject s3:ListBucket	應用程式
UpdateEnvironment	kms:DescribeKey	環境

AWS 大型主機現代化的原則條件金鑰

支援服務特定政策條件金鑰：是

管理員可以使用 AWS JSON策略來指定誰可以存取什麼內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Condition 元素 (或 Condition 區塊) 可讓您指定使陳述式生效的條件。Condition 元素是選用項目。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵， AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值，請使用邏輯OR運算來 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，只有在IAM使用者名稱標記資源時，您才可以授與IAM使用者存取資源的權限。如需詳細資訊，請參閱《IAM使用指南》中的IAM政策元素：變數和標籤。

AWS 支援全域條件金鑰和服務特定條件金鑰。若要查看所有 AWS 全域條件索引鍵，請參閱《使用指南》中的AWS 全域條件內IAM容索引鍵。

下列條件索引鍵是 AWS 大型主機現代化的特定條件

```
m2:EngineType
            m2:InstanceType
```

若要查看 AWS 大型主機現代化條件金鑰清單，請參閱服務授權參考資料中的AWS 大型主機現代化的條件金鑰。若要瞭解您可以使用條件金鑰的動作和資源，請參閱AWS 大型主機現代化所定義的動作。

若要檢視大型主機現代化身分 AWS 型原則的範例，請參閱。大型主機現代化的身分識別原則範例 AWS

AWS 大型主機現代化中的存取控制清單 (ACLs)

支持ACLs：無

存取控制清單 (ACLs) 控制哪些主參與者 (帳戶成員、使用者或角色) 具有存取資源的權限。ACLs類似於以資源為基礎的策略，雖然它們不使用JSON政策文件格式。

以屬性為基礎的存取控制 (ABAC) 搭配 AWS 大型主機現代化

支援 ABAC (策略中的標籤): 是

以屬性為基礎的存取控制 (ABAC) 是一種授權策略，可根據屬性定義權限。在中 AWS，這些屬性稱為標籤。您可以將標籤附加至IAM實體 (使用者或角色) 和許多 AWS 資源。標記實體和資源是的第一步ABAC。然後，您可以設計ABAC策略，以便在主參與者的標籤與他們嘗試存取的資源上的標籤相符時允許作業。

ABAC在快速成長的環境中很有幫助，並且有助於原則管理變得繁瑣的情況。

如需根據標籤控制存取，請使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 條件索引鍵，在政策的條件元素中，提供標籤資訊。

如果服務支援每個資源類型的全部三個條件金鑰，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件金鑰，則值為 Partial。

如需有關的詳細資訊ABAC，請參閱什麼是ABAC？ 在《IAM使用者指南》中。若要檢視包含設定步驟的自學課程ABAC，請參閱《使用指南》中的〈使用以屬性為基礎的存取控制 (ABAC) IAM 〉。

使用臨時登入資料搭配 AWS 大型主機現代化

支援臨時憑證：是

當您使用臨時憑據登錄時，某些 AWS 服務 不起作用。如需其他資訊，包括哪些 AWS 服務 與臨時登入資料搭配使用 AWS 服務 ，請參閱《IAM使用指南》IAM中的使用方式。

如果您使用除了使用者名稱和密碼以外的任何方法登入，則您正在 AWS Management Console 使用臨時認證。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取時，該程序會自動建立臨時認證。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需有關切換角色的詳細資訊，請參閱《IAM使用者指南》中的〈切換到角色 (主控台)〉。

您可以使用 AWS CLI 或手動建立臨時認證 AWS API。然後，您可以使用這些臨時登入資料來存取 AWS。 AWS 建議您動態產生臨時登入資料，而不是使用長期存取金鑰。如需詳細資訊，請參閱IAM。

AWS 大型主機現代化的轉寄存取工作階段

支援轉寄存取工作階段 (FAS)：是

當您使用使用IAM者或角色在中執行動作時 AWS，您會被視為主參與者。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS會使用主參與者呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。FAS只有當服務收到需要與其他 AWS 服務 資源互動才能完成的請求時，才會發出請求。在此情況下，您必須具有執行這兩個動作的許可。有關提出FAS請求時的策略詳細信息，請參閱轉發訪問會話。

> **Important**
>
> 這些權杖可讓 AWS 大型主機現代化存取客戶資料，而無需明確協議；例如， AWS 大型主機現代化可以從 Amazon S3 儲存貯體部署應用程式構件與相關業務資料，而無需取得客戶明確的許可。您可能需要相應地更新任何合規性文件。

AWS 大型主機現代化的服務角色

支援服務角色：是

服務角色是服務假定代表您執行動作的IAM角色。IAM管理員可以從中建立、修改和刪除服務角色IAM。如需詳細資訊，請參閱《IAM使用指南》 AWS 服務中的建立角色以將權限委派給

AWS 大型主機現代化支援活動掛接的服務角色 (交易/工作異常結束或完成等)。

> **Warning**
>
> 變更服務角色的權限可能會中斷 AWS 大型主機現代化功能。只有在 AWS 大型主機現代化提供指引時，才能編輯服務角色。

選擇 AWS 大型主機現代化中的IAM角色

如果您之前已經建立IAM了在 Amazon 上執行的應用程式EC2可以擔任的角色，則可以在建立啟動範本或啟動組態時選擇此角色。 AWS 大型主機現代化提供您可供選擇的角色清單。建立這些角色時，請務必將限制存取權的最低權限IAM原則建立關聯至應用程式所需的特定API呼叫。如需詳細資訊，請參閱Amazon EC2Auto Scaling 使用者指南中針對在 Amazon EC2 執行個體上執行的應用程式IAM角色。

AWS 大型主機現代化的服務連結角色

支援服務連結角色：是

服務連結角色是一種連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM管理員可以檢視 (但無法編輯服務連結角色) 的權限。

如需建立或管理 AWS 大型主機現代化服務連結角色的詳細資訊，請參閱。使用 AWS Mainframe Modernization的服務連結角色

如需有關建立或管理服務連結角色的詳細資訊，請參閱使用IAM的AWS 服務。在表格中尋找服務，其中包含服務連結角色欄中的 Yes。選擇是連結，以檢視該服務的服務連結角色文件。

大型主機現代化的身分識別原則範例 AWS

根據預設，使用者和角色沒有建立或修改 AWS 大型主機現代化資源的權限。他們也無法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或執行工作 AWS API。若要授與使用者對所需資源執行動作的權限，IAM管理員可以建立IAM策略。然後，系統管理員可以將IAM原則新增至角色，使用者可以擔任這些角色。

若要瞭解如何使用這些範例原則文件來建立以IAM身分識別為基礎的JSON策略，請參閱使用指南中的IAM建立IAM策略。

如需有關 AWS 大型主機現代化所定義之動作和資源類型的詳細資訊，包括各資源類型的格式，請參閱服務授權參考中大型AWS 主機現代化的動作、資源和條件索引鍵。ARNs

主題

- 政策最佳實務
- 使用 AWS 大型主機現代化主控台
- 允許使用者檢視他們自己的許可

政策最佳實務

以身分識別為基礎的原則可決定使用者是否可以建立、存取或刪除您帳戶中的 AWS 大型主機現代化資源。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管原則並邁向最低權限權限 — 若要開始授與使用者和工作負載的權限，請使用可授與許多常見使用案例權限的AWS 受管理原則。它們可用在您的 AWS 帳戶. 建議您透過定義特定於您使用案例的 AWS 客戶管理政策，進一步降低使用權限。如需詳細資訊，請參閱AWS 《IAM使用指南》中針對工作職能的AWS 受管理策略或受管理的策略。
- 套用最低權限權限 — 當您使用原則設定權限時，IAM只授與執行工作所需的權限。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需有關使用套用權限IAM的詳細資訊，請參閱《使用指南》IAM中的IAM《策略與權限》。
- 使用IAM策略中的條件進一步限制存取 — 您可以在策略中新增條件，以限制對動作和資源的存取。例如，您可以撰寫政策條件，以指定必須使用傳送所有要求SSL。您也可以使用條件來授與對服務動作的存取權 (如透過特定) 使用這些動作 AWS 服務，例如 AWS CloudFormation。如需詳細資訊，請參閱《IAM使用指南》中的IAMJSON策略元素：條件。
- 使用 IAM Access Analyzer 驗證您的原IAM則，以確保安全和功能性的權限 — IAM Access Analyzer會驗證新的和現有的原則，以便原則遵循IAM原則語言 (JSON) 和IAM最佳做法。IAMAccess

Analyzer 提供超過 100 項原則檢查和可採取動作的建議，協助您撰寫安全且功能正常的原則。如需詳細資訊，請參閱IAM使IAM用指南中的存取分析器原則驗證。

- 需要多因素驗證 (MFA) — 如果您的案例需要使IAM用者或 root 使用者 AWS 帳戶，請開啟以取得額外MFA的安全性。若要在呼叫API作業MFA時需要，請在原則中新增MFA條件。如需詳細資訊，請參閱《IAM使用指南》中的 < 設定MFA受保護的API存取 > 。

如需有關中最佳作法的詳細資訊IAM，請參閱《IAM使用指南》IAM中的「安全性最佳作法」。

使用 AWS 大型主機現代化主控台

若要存取 AWS 大型主機現代化主控台，您必須擁有最少一組權限。這些權限必須允許您列出和檢視有關. AWS AWS 帳戶如果您建立比最基本必要許可更嚴格的身分型政策，則對於具有該政策的實體 (使用者或角色) 而言，主控台就無法如預期運作。

您不需要為只對 AWS CLI 或撥打電話的使用者允許最低主控台權限 AWS API。相反地，只允許存取符合他們嘗試執行之API作業的動作。

若要確保使用者和角色仍可使用 AWS 大型主機現代化主控台，請同時將大型主 AWS 機現代化`ConsoleAccess`或`ReadOnly` AWS 受管理的原則附加至實體。如需詳細資訊，請參閱《使用指南》中的〈將權限新增至IAM使用者〉。

允許使用者檢視他們自己的許可

此範例顯示如何建立原則，讓使IAM用者檢視附加至其使用者身分識別的內嵌和受管理原則。此原則包含在主控台上或以程式設計方式使用或完成此動作的 AWS CLI 權限 AWS API。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
```

```
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

疑難排解 AWS 大型主機現代化身分識別與存取

使用下列資訊可協助您診斷並修正使用 AWS 大型主機現代化和時可能會遇到的常見問題。IAM

主題

- 我沒有授權執行 iam：PassRole
- 我想要允許我以外的人員存取我 AWS 帳戶 的 AWS 大型主機現代化資源

我沒有授權執行 iam：PassRole

如果您收到未獲授權執行`iam:PassRole`動作的錯誤訊息，則必須更新您的原則，以允許您將角色傳遞 AWS 給大型主機現代化。

有些 AWS 服務 允許您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

當名為的使用IAM者`marymajor`嘗試使用主控台在 AWS 大型主機現代化中執行動作時，就會發生下列範例錯誤。但是，動作請求服務具備服務角色授予的許可。Mary 沒有將角色傳遞至該服務的許可。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在這種情況下，Mary 的政策必須更新，允許她執行 `iam:PassRole` 動作。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

我想要允許我以外的人員存取我 AWS 帳戶 的 AWS 大型主機現代化資源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。對於支援以資源為基礎的政策或存取控制清單 (ACLs) 的服務，您可以使用這些政策授與人員存取您資源的權限。

如需進一步了解，請參閱以下內容：

- 若要瞭解 AWS 大型主機現代化是否支援這些功能，請參閱。AWS 大型主機現代化如何搭配運作 IAM
- 若要瞭解如何提供您所擁有資源 AWS 帳戶 的存取權，請參閱《IAM使用者指南》中 AWS 帳戶 的〈提供存取權給其他IAM使用者〉。
- 若要瞭解如何將資源存取權提供給第三方 AWS 帳戶，請參閱《IAM使用指南》中的提供第三方 AWS 帳戶 擁有的存取權。
- 若要瞭解如何透過身分聯盟提供存取權，請參閱使用指南中的提供對外部驗證使用IAM者的存取權 (身分聯合)。
- 若要瞭解針對跨帳號存取使用角色與以資源為基礎的政策之間的差異，請參閱《使用IAM者指南》IAM中的〈跨帳號資源存取〉。

使用 AWS Mainframe Modernization的服務連結角色

AWS Mainframe Modernization 使用 AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是直接連結至 AWS Mainframe Modernization的唯一IAM角色類型。服務連結角色由預先定義， AWS Mainframe Modernization 並包含服務代表您呼叫其他服 AWS 務所需的所有權限。

服務連結角色可讓您 AWS Mainframe Modernization 更輕鬆地設定，因為您不必手動新增必要的權限。 AWS Mainframe Modernization 定義其服務連結角色的權限，除非另有定義，否則只 AWS Mainframe Modernization 能擔任其角色。定義的權限包括信任原則和權限原則，而且該權限原則無法附加至任何其他IAM實體。

您必須先刪除服務連結角色的相關資源，才能將其刪除。這樣可以保護您的 AWS Mainframe Modernization 資源，因為您無法不小心移除存取資源的權限。

如需支援服務連結角色之其他服務的相關資訊，請參閱使用的AWS 服務，IAM並在服務連結角色欄中尋找具有是的服務。選擇具有連結的是，以檢視該服務的服務連結角色文件。

AWS Mainframe Modernization的服務連結角色許可

AWS Mainframe Modernization 使用名為的服務連結角色 AWSServiceRoleForAWSM2— 將網路配置為連線至您的網路VPC並存取檔案系統等資源。

AWSServiceRoleForAWSM2 個服務連結角色會信任下列服務擔任該角色：

- `m2.amazonaws.com`

名為的角色權限原則 AWSM2ServicePolicy AWS Mainframe Modernization 允許對指定的資源完成下列動作：

- 為 AWS Mainframe Modernization 環境建立、刪除、描述和附加許可至 Amazon EC2 網路界面，以建立與客戶的連線能力VPC。
- 從 Elastic Load Balancing 註冊或取消註冊項目，這是客戶連接到 AWS Mainframe Modernization 環境的方式。
- 描述 Amazon EFS 或 Amazon FSx 文件系統（如果使用的話）。
- 從執行階段環境向客戶 CloudWatch 發出指標。

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "ec2:DescribeSubnets",
    "ec2:CreateNetworkInterface",
    "ec2:DeleteNetworkInterface",
    "ec2:DescribeNetworkInterfaces",
    "ec2:CreateNetworkInterfacePermission",
    "ec2:ModifyNetworkInterfaceAttribute"
   ],
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "elasticfilesystem:DescribeMountTargets"
   ],
   "Resource": "*"
```

```
  },
  {
   "Effect": "Allow",
   "Action": [
    "elasticloadbalancing:RegisterTargets",
    "elasticloadbalancing:DeregisterTargets"
   ],
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "fsx:DescribeFileSystems"
   ],
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "cloudwatch:PutMetricData"
   ],
   "Resource": "*",
   "Condition": {
    "StringEquals": {
     "cloudwatch:namespace": [
      "AWS/M2"
     ]
    }
   }
  }
 ]
}
```

您必須設定權限，才能允許IAM實體 (例如使用者、群組或角色) 建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱IAM使用指南中的服務連結角色權限。

為 AWS Mainframe Modernization建立服務連結角色

您不需要手動建立一個服務連結角色。當您在 AWS Management Console、或中建立執行階段環境時 AWS CLI， AWS Mainframe Modernization 會為您建立服務連結角色。 AWS API

若您刪除此服務連結角色，之後需要再次建立，您可以在帳戶中使用相同程序重新建立角色。當您建立執行階段環境時，請再次為您建 AWS Mainframe Modernization 立服務連結角色。

為 AWS Mainframe Modernization編輯服務連結角色

AWS Mainframe Modernization 不允許您編輯 AWSServiceRoleForAWSM 2 個服務連結角色。因為有各種實體可能會參考服務連結角色，所以您無法在建立角色之後變更角色名稱。但是，您可以使用編輯角色的描述IAM。如需詳細資訊，請參閱IAM使用指南中的編輯服務連結角色。

為 AWS Mainframe Modernization刪除服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。然而，在手動刪除服務連結角色之前，您必須先清除資源。

> **Note**
>
> 當您嘗試刪除資源時，如果 AWS Mainframe Modernization 服務正在使用此角色，則刪除可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試操作。

要刪除由 AWSServiceRoleForAWSM 2 使用的 AWS Mainframe Modernization 資源

- 刪除中的執行階段環境 AWS Mainframe Modernization。在刪除環境本身之前，請務必先從環境中刪除應用程式。

若要使用手動刪除服務連結角色 IAM

使用IAM主控台 AWS CLI、或刪除 AWSServiceRoleForAWSM 2 AWS API 個服務連結角色。如需詳細資訊，請參閱IAM使用指南中的刪除服務連結角色。

AWS Mainframe Modernization 服務連結角色的支援區域

AWS Mainframe Modernization 支援在所有提供服務的區域中使用服務連結角色。如需詳細資訊，請參閱 AWS 區域與端點。

適用於 AWS 大型主機現代化的合規性驗證

協力廠商稽核員會評估 AWS 大型主機現代化的安全性與合規性，做為多項合規計畫的一部分。 AWS 這些措施包括 SOC PCIRAMP, 美聯儲HIPAA, 和其他.

如需特定規範計劃範圍內的 AWS 服務清單，請參閱合規計劃AWS服務範圍方案)。如需一般資訊，請參閱 AWS 合規計劃 。

您可以使用下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱下載中的報告中的 AWS Artifact。

您在使用 AWS 大型主機現代化時的合規責任取決於資料的敏感度、公司的合規目標，以及適用的法律和法規。 AWS 提供下列資源以協助遵循法規：

- 安全與合規快速入門指南：這些部署指南討論架構考量，並提供在 AWS上部署以安全及合規為重心之基準環境的步驟。
- HIPAA安全性與合規性架構白皮書 — 本白皮書說明公司如何使用建立HIPAA符合標準的應 AWS 用程式。
- AWS 合規資源AWS — 此工作簿和指南集合可能適用於您的產業和所在地。
- 使用AWS Config 開發人員指南中的規則評估資源 — AWS Config；評估您的資源配置如何符合內部實踐，業界準則和法規。
- AWS Security Hub— 此 AWS 服務提供安全狀態的全面檢視，協助您檢查您 AWS 是否符合安全性產業標準和最佳做法。

AWS 大型主機現代化的韌性

AWS 全球基礎架構是圍繞區 AWS 域和可用區域建立的。區域提供多個分開且隔離的實際可用區域，並以低延遲、高輸送量和高度備援網路連線相互連結。透過可用區域，您可以設計與操作的應用程式和資料庫，在可用區域之間自動容錯移轉而不會發生中斷。可用區域的可用性、容錯能力和擴展能力，均較單一或多個資料中心的傳統基礎設施還高。

如需區域和可用區域的相關 AWS 資訊，請參閱AWS 全域基礎結構。

基礎結構安全 AWS Mainframe Modernization

作為託管服務， AWS Mainframe Modernization 受到 AWS 全球網絡安全的保護。有關 AWS 安全服務以及如何 AWS 保護基礎結構的詳細資訊，請參閱AWS 雲端安全 若要使用基礎架構安全性的最佳做法來設計您的 AWS 環境，請參閱安全性支柱架構良 AWS 好的架構中的基礎結構保護。

您可以使用 AWS 已發佈的API呼叫透 AWS Mainframe Modernization 過網路存取。使用者端必須支援下列專案：

- 傳輸層安全性 (TLS)。我們需要 TLS 1.2 並推薦 TLS 1.3。
- 具有完美前向保密（ ）的密碼套件，例如（短暫的迪菲-赫爾曼PFS）或DHE（橢圓曲線短暫迪菲-赫爾曼）。ECDHE現代系統(如 Java 7 和更新版本)大多會支援這些模式。

此外，請求必須使用存取金鑰 ID 和與IAM主體相關聯的秘密存取金鑰來簽署。或者，您可以透過 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

使 AWS Mainframe Modernization 用 AWS PrivateLink 介面端點存取

您可以使 AWS PrivateLink 用在VPC和之間建立私人連線 AWS Mainframe Modernization。您可以 AWS Mainframe Modernization 像在您的一樣訪問VPC，而無需使用 Internet 網關，NAT設備，VPN 連接或 AWS Direct Connect 連接。您中的執行個體VPC不需要公用 IP 位址即可存取 AWS Mainframe Modernization。

您可以建立由 AWS PrivateLink提供支援的介面端點來建立此私有連線。我們會在您為介面端點啟用的每個子網中建立端點網路介面。這些是請求者管理的網路介面，可作為目的地為 AWS Mainframe Modernization之流量的進入點。

如需詳細資訊，請參閱AWS PrivateLink 指南 AWS PrivateLink中的 AWS 服務 透過存取。

的注意事項 AWS Mainframe Modernization

設定的介面端點之前 AWS Mainframe Modernization，請先檢閱AWS PrivateLink 指南中的考量事項。

AWS Mainframe Modernization 支援透過介面端點呼叫其所有API動作。

建立的介面端點 AWS Mainframe Modernization

您可以建立 AWS Mainframe Modernization 使用 Amazon VPC 主控台或 AWS Command Line Interface (AWS CLI) 的介面端點。如需詳細資訊，請參閱《AWS PrivateLink 指南》中的建立介面端點。

建立 AWS Mainframe Modernization 使用下列服務名稱的介面端點：

```
com.amazonaws.region.m2
```

如果您DNS為介面端點啟用 private，您可以 AWS Mainframe Modernization 使用其預設地區DNS名稱提出API要求。例如：`m2.us-east-1.amazonaws.com`。

為您的介面端點建立端點政策

端點策略是您可以附加到介面端點的IAM資源。預設端點策略允許 AWS Mainframe Modernization 透過介面端點進行完整存取。若要控制 AWS Mainframe Modernization 從您的允許存取VPC，請將自訂端點原則附加到介面端點。

端點政策會指定以下資訊：

- 可以執行動作 (AWS 帳戶、使用者和IAM角色) 的主參與者。
- 可執行的動作。
- 可供執行動作的資源。

如需詳細資訊，請參閱《AWS PrivateLink 指南》中的使用端點政策控制對服務的存取。

範例： AWS Mainframe Modernization 處理行動的VPC端點策略

以下是自訂端點政策的範例。將此政策附加至介面端點後，此政策會針對所有資源上的所有主體，授予列出的 AWS Mainframe Modernization 動作的存取權限。

```
//Example of an endpoint policy where access is granted to the
//listed AWS Mainframe Modernization actions for all principals on all resources
{"Statement": [
      {"Principal": "*",
         "Effect": "Allow",
         "Action": [
            "m2:ListApplications",
            "m2:ListEnvironments",
            "m2:ListDeployments"
         ],
         "Resource":"*"
      }
   ]
}

//Example of an endpoint policy where access is denied to all the
//AWS Mainframe Modernization CREATE actions for all principals on all resources
{"Statement": [
      {"Principal": "*",
         "Effect": "Deny",
         "Action": [
            "m2:Create*"
```

```
            ],
            "Resource":"*"
        }
    ]
}
```

監控 AWS 大型主機現代化

監控是維持 AWS 大型主機現代化和其他解決方案的可靠性、可用性和效能的重要組成部分。AWSAWS提供下列監視工具來觀看 AWS 大型主機現代化、回報發生錯誤時，並在適當時採取自動動作：

- Amazon 會即時 CloudWatch監控您的 AWS 資源和執行 AWS 的應用程式。您可以收集和追蹤指標、建立自訂儀板表，以及設定警示，在特定指標達到您指定的閾值時通知您或採取動作。例如，您可以 CloudWatch 追蹤 Amazon EC2 執行個體的CPU使用情況或其他指標，並在需要時自動啟動新執行個體。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南。
- Amazon CloudWatch 日誌可讓您從 Amazon EC2 執行個體和其他來源監控 CloudTrail、存放和存取日誌檔。 CloudWatch 記錄檔可以監控記錄檔中的資訊，並在符合特定臨界值時通知您。您也可以將日誌資料存檔在高耐用性的儲存空間。如需詳細資訊，請參閱 Amazon CloudWatch 日誌使用者指南。
- AWS CloudTrail擷取您帳戶或代表您的 AWS 帳戶發出的API呼叫和相關事件，並將日誌檔傳送到您指定的 Amazon S3 儲存貯體。您可以識別呼叫的使用者和帳戶 AWS、進行呼叫的來源 IP 位址，以及呼叫發生的時間。如需詳細資訊，請參閱《AWS CloudTrail 使用者指南》https://docs.aws.amazon.com/awscloudtrail/latest/userguide/。

使用 Amazon 監控 AWS 大型主機現代化 CloudWatch

您可以使用 AWS 來監控大型主機現代化 CloudWatch，這會收集原始資料，並將其處理成可讀且近乎即時的指標。這些統計資料會保留 15 個月，以便您存取歷史資訊，並更清楚 Web 應用程式或服務的執行效能。您也可以設定留意特定閾值的警示，當滿足這些閾值時傳送通知或採取動作。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南。

下表列出 AWS 大型主機現代化的度量和維度。這些指標的命名空間為AWS/M2。

執行環境測量結

指標	描述
CPUUtilization	環境中執行個體的使CPU用率。 尺寸:environmentId 單位：百分比

指標	描述
	有效統計：平均、最小值、最大值
InboundNetworkThroughput	環境中執行個體的輸入網路輸送量。 尺寸:environmentId 單位：位元組/秒 有效統計：平均、最小值、最大值
MemoryUtilization	環境中執行個體的記憶體使用率。 尺寸:environmentId 單位：百分比 有效統計：平均、最小值、最大值
OutboundNetworkThroughput	環境中執行個體的輸出網路輸送量。 尺寸:environmentId 單位：位元組/秒 有效統計：平均、最小值、最大值

應用程式指標

指標	描述
BatchJobCompletedCount	時間間隔內已完成的工作數目。 此指標適用於微型焦點和 AWS 藍光時代 3.7.0 及更新版本。 尺寸:applicationId 單位：計數

指標	描述
	有效的統計資訊：總和
BatchJobFailedCount	時間間隔內失敗的工作數目。 此指標適用於微型焦點和 AWS 藍光時代 3.7.0 及更新版本。 尺寸:applicationId 單位：計數 有效的統計資訊：總和
JvmMemoryFree	Java 虛擬機器目前未使用的可用記憶體數量。 此指標僅適用於 AWS 藍光時代執行階段引擎。它適用於 AWS 藍光時代 3.7.0 及更高版本。 尺寸:applicationId 單位：位元組 有效統計：平均、最小值、最大值
JvmMemoryMax	Java 虛擬機器允許的最大記憶體容量。 此指標僅適用於 AWS 藍光時代執行階段引擎。它適用於 AWS 藍光時代 3.7.0 及更高版本。 尺寸:applicationId 單位：位元組 有效統計：平均、最小值、最大值

指標	描述
JvmMemoryUsed	Java 虛擬機器使用中的記憶體數量。 此指標僅適用於 AWS 藍光時代執行階段引擎。它適用於 AWS 藍光時代 3.7.0 及更高版本。 尺寸:applicationId 單位：位元組 有效統計：平均、最小值、最大值
ProcessesActiveCount	正在處理請求之並行服務執行處理作業的有效數目。 此指標僅適用於 Micro Focus 執行階段引擎。 尺寸:applicationId 單位：計數 有效的統計資訊：總和
SessionCount	應用程式的HTTP工作階段數目。 此指標僅適用於 AWS 藍光時代執行階段引擎。它適用於 AWS 藍光時代 3.7.0 及更高版本。 尺寸:applicationId 單位：計數 有效統計：平均、最小值、最大值

指標	描述
SharedMemoryFree	企業伺服器可用來儲存執行交易和工作所需的所有資訊的記憶體。 此指標僅適用於 Micro Focus 執行階段引擎。 尺寸:applicationId 單位：計數 有效統計：平均、最小值、最大值
ThreadActiveCount	正在處理要求的引擎執行緒數目。 此指標僅適用於 AWS 藍光時代執行階段引擎。它適用於 AWS 藍光時代 3.7.0 及更高版本。 尺寸:applicationId 單位：計數 有效統計：平均、最小值、最大值
TransactionCompletedCount	時間間隔內確認的交易數目。 此指標適用於微型焦點和 AWS 藍光時代 3.7.0 及更新版本。 尺寸:applicationId 單位：計數 有效的統計資訊：總和

指標	描述
TransactionFailedCount	時間間隔內失敗的交易數目。 此指標適用於微型焦點和 AWS 藍光時代 3.7.0 及更新版本。 尺寸:applicationId 單位：計數 有效的統計資訊：總和
TransactionResponseTime	從使用者傳送要求到應用程式指出要求已完成為止的時間長度。 此指標適用於微型焦點和 AWS 藍光時代 3.7.0 及更新版本。 尺寸:applicationId 單位：毫秒 有效統計：平均、最小值、最大值

維度

維度	描述
applicationId	此維度會依 ID 將量度篩選為已識別的應用程式。
environmentId	此維度會依 ID 將度量篩選至已識別的環境。

使用下列方式記錄 AWS 大型主機現代化呼API叫 AWS CloudTrail

AWS 大型主機現代化與服務整合在一起 AWS CloudTrail，可提供使用者、角色或服務在大型主機現代化中所採取的動作記錄的 AWS AWS 服務。 CloudTrail 將 AWS 大型主機現代化的所有要API求擷取為

活動。擷取的呼叫包括來自 AWS 大型主機現代化主控台的呼叫，以及對大型主機現代化作業的程式碼呼叫。 AWS API如果您建立追蹤，您可以啟用持續交付 CloudTrail 事件到 Amazon S3 儲存貯體，包括 AWS 大型主機現代化的事件。如果您未設定追蹤，您仍然可以在 [事件歷程記錄] 中檢視 CloudTrail 主控台中最近的事件。使用收集的資訊 CloudTrail，您可以判斷對 AWS 大型主機現代化提出的要求、提出要求的 IP 位址、提出要求的人員、提出要求的時間，以及其他詳細資訊。

若要進一步了解 CloudTrail，請參閱使AWS CloudTrail 用者指南。

AWS 大型主機現代化資訊 CloudTrail

CloudTrail 在您創建 AWS 帳戶時，您的帳戶已啟用。當活動在 AWS 大型主機現代化中發生時，該活動會與事件歷史記錄中的其他 AWS 服務 CloudTrail 事件一起記錄在事件中。您可以在帳戶中查看，搜索和下載最近的事 AWS 件。如需詳細資訊，請參閱檢視具有事 CloudTrail 件記錄的事件。

如需 AWS 帳戶中持續記錄事件 (包括大 AWS 型主機現代化的活動)，請建立追蹤。追蹤可 CloudTrail 將日誌檔交付到 Amazon S3 儲存貯體。根據預設，當您在主控台中建立追蹤時，追蹤會套用至所有 AWS區域。追蹤記錄來自 AWS 分區中所有區域的事件，並將日誌檔傳送到您指定的 Amazon S3 儲存貯體。此外，您還可以設定其他 AWS 服務，以進一步分析 CloudTrail 記錄中收集的事件資料並採取行動。如需詳細資訊，請參閱下列內容：

- 建立追蹤的概觀
- CloudTrail 支援的服務與整合
- 設定 Amazon SNS 通知 CloudTrail
- 從多個區域接收 CloudTrail 記錄檔
- 從多個帳戶接收 CloudTrail 日誌文件

所有 AWS 大型主機現代化動作均由記錄， CloudTrail 並記錄在大型主機現代化參考資料中AWS 。API例如，呼叫`CreateEnvironment`和`CreateDeployment`動作會`CreateApplication`在 CloudTrail 記錄檔中產生項目。

每一筆事件或日誌專案都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 該請求是使用根使用者還是使用者憑證提出。
- 提出該請求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 請求是否由其他 AWS 服務提出。

如需詳細資訊，請參閱CloudTrail userIdentity 元素。

瞭解 AWS 大型主機現代化記錄檔項目

追蹤是一種組態，可讓事件以日誌檔的形式傳遞到您指定的 Amazon S3 儲存貯體。 CloudTrail 記錄檔包含一或多個記錄項目。事件代表來自任何來源的單一請求，包括有關請求的操作，動作的日期和時間，請求參數等信息。 CloudTrail 日誌文件不是公共API調用的有序堆棧跟踪，因此它們不會以任何特定順序顯示。

下列範例顯示示範`CreateApplication`動作的 CloudTrail 記錄項目。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAII6WZTHGYAEXAMPLE",
        "arn": "arn:aws:sts::444455556666:assumed-role/Admin/Mary_Major",
        "accountId": "444455556666",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAII6WZTHGYAEXAMPLE",
                "arn": "arn:aws:iam::444455556666:role/Admin",
                "accountId": "444455556666",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-06-01T20:38:22Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-06-01T20:40:39Z",
    "eventSource": "m2.amazonaws.com",
    "eventName": "CreateApplication",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "72.21.196.65",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:91.0) Gecko/20100101
 Firefox/91.0",
    "requestParameters": {
        "clientToken": "1abc23de-f45g-6789-h01i-jkl2m3456789",
        "name": "MyApp",
        "description": "",
```

```
        "engineType": "microfocus",
        "definition": {
            "content": "{}"
        },
        "tags": {}
    },
    "responseElements": {
        "applicationVersion": 1,
        "Access-Control-Expose-Headers": "x-amzn-RequestId,x-amzn-ErrorType,x-amzn-
ErrorMessage,Date",
        "applicationArn": "arn:aws:m2:us-east-1:444455556666:app/
lsfhmwhw7fffrosff2lncwqcua",
        "applicationId": "lsfhmwhw7fffrosff2lncwqcua"
    },
    "requestID": "36982d38-fcde-4bfe-a89a-7bd78d43c926",
    "eventID": "d7f0fc36-46ae-4157-9a79-c79f385fda98",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "444455556666",
    "eventCategory": "Management"
}
```

AWS 大型主機現代化中的疑難排解

使用本節中的資訊可協助您針 AWS 對使用 AWS Blu Age 和 Micro Focus 引擎的大型主機現代化應用程式和執行階段環境中的常見錯誤進行疑難排解。

主題

- 疑難排解錯誤：等待資料集名稱解除鎖定時逾時
- 疑難排解錯誤：無法存取應用程式 URL
- 疑難排解：無法從主控台開啟 AWS 藍光深入解析
- 疑難排解錯誤：環境狀況不良
- 疑難排解微焦點的授權問題

疑難排解錯誤：等待資料集名稱解除鎖定時逾時

本頁說明當您看到環境中的另一個應用程式正在鎖定共用資料集時，如何解決錯誤。

- 引擎： AWS 藍色時代
- 組件：布魯薩姆

如果您在使用 AWS Blu Age 引擎的 AWS 大型主機現代化應用程式的 Amazon CloudWatch 日誌中看到此錯誤，並且在具有高可用性模式的環境中執行，則表示另一個應用程式正在共用資料集上鎖。通常情況下，如果其他應用程式當機或以其他方式失敗且未解除鎖定，就會發生這種情況。

尋找失敗的應用程式，並檢查它是否使用錯誤訊息中提到的相同資料集。檢查應用程式是否在具有高可用性模式的執行階段環境中執行。引發逾時例外狀況的應用程式無法繼續，並會顯示`Failed`狀態。

常見原因

應用程式`example-app-1`嘗試鎖定寫入作業`example-record-1`的記錄。此作業會在擁有的資料集`example-dataset-1`上建立鎖定`example-record-1`，也會建立鎖定`example-record-1`本身。現在，`example-app-2`另一個應用程序嘗試鎖定相同的記錄`example-record-1`。資料集和記錄已鎖定，因此請`example-app-2`等待鎖定釋放。如果`example-app-1`當機，資料集上的保留鎖定`example-dataset-1`仍然存在，這會導`example-app-2`致取消其寫入嘗試並引發逾時例外狀況。這種死結情況可以防止所有應用程序到達`example-dataset-1`。

解析度

若要立即解決此情況，您可以強制釋放鎖定。為了防止 future 發生類似情況，您可以配置兩個控制 Blusam auto 修復機制的參數。

強制鎖定釋放

Blusam 鎖定管理器使用 Amazon ElastiCache （RedisOSS）在應用程序之間提供共享鎖。若要解除中的鎖定 ElastiCache，請使用 Redis CLI 公用程式。您無法刪除個別記錄鎖定。您必須移除擁有資料集中的所有鎖定。請完成下列步驟：

1. ElastiCache 使用以下命令 Connect 到您的：

   ```
   redis-cli -h hostname -p port
   ```

   您可以在 ElastiCache 控制台中找到您 ElastiCache 的詳細信息，位於 https://console.aws.amazon.com/elasticache/。
2. 輸入您的密碼。
3. 輸入您要執行的命令，如下所示：

Command	用途
KEYS *	獲取所有現有的密鑰。
KEYS *YOUR_DATASET_NAME	取得資料集鎖定金鑰。
DEL THE_RETURNED_KEY	刪除資料集鎖定。
FLUSHDB	清理整個 Redis 的。 **Warning** Redis 快取中的所有資料都會遺失。如果 Redis 用於其他目的，例如處理 http 會話，則可能不想使 FLUSHDB用。

配置布魯桑自 auto 修復機制

Blusam 鎖定管理器包括一個 auto 修復機制，以防止數據集或記錄死鎖。您可以在應用程式定義 (application-main.yml) 中調整下列參數，以配置 auto 修復機制：

- locksDeadTime: 是指應用程序可以保持鎖的最長時間。當這段時間過去時，鎖定聲明過期並立即釋放。locksDeadTime值以毫秒為單位，預設值為 1000。
- locksCheck：定義用於檢查鎖定的 Blusam 鎖定管理員策略。所有 Blusam 鎖定 ElastiCache 都有時間戳，並且有到期時間。locksCheck參數值決定是否移除過期的鎖定。
  - off：任何時候都不會執行任何檢查。可能會發生死結。(不建議使用)
  - reboot：在大型主機現代化執行階段環境中執行的 AWS 大型主機現代化應用程式執行個 AWS 體啟動或重新啟動時，會執行檢查。所有過期的鎖定都會立即釋放。(預設值)
  - timeout：在大型主機現代化執行階段環境中執行的 AWS 大型主機現代化應用程式執行個 AWS 體啟動或重新啟動時，或在嘗試鎖定資料集期間逾時到期時，會執行檢查。過期的鎖定會立即釋放。

如需 AWS Blu Age 應用程式之應用程式定義的詳細資訊，請參閱AWS 藍光時代的應用程序定義。

布魯薩姆鎖管理器

在使用高可用性模式的 AWS 大型主機現代化執行階段環境中， AWS Blu Age 應用程式可能會部署多次。對於處理 Blusam 資料集的應用程式，可能會發生並行存取問題。Blusam 鎖管理器確保數據完整性，並通過使用應用程序之間提供共享鎖來管理對記錄和數據集的讀寫訪問。 ElastiCache這種機制允許多個應用程序同時讀取記錄，並確保一次只有一個應用程序寫入記錄。

寫鎖

若要更新或刪除特定記錄，應用程式必須先鎖定擁有該記錄的資料集，然後鎖定記錄本身。鎖定記錄時，會釋放資料集鎖定，同一資料集中的其他記錄可供使用。更新或刪除作業完成時，會釋放保留的記錄鎖定。一次只有一個應用程式可以更新記錄，如果定義的應用程式原則允許等待釋放，則會封鎖其他應用程式讀取或寫入，直到釋放鎖定為止。

讀取鎖

只要記錄或資料集上沒有寫入鎖定，多個應用程式就可以同時讀取相同的記錄。若要鎖定寫入作業的記錄，必須釋放所有讀取鎖定。

> **Note**
>
> Blusam 鎖定管理器使用相同的鎖定機制處理來自給定應用程序中的多個線程的訪問。

疑難排解錯誤：無法存取應用程式 URL

本頁說明當您無法存取執行中的 AWS 大型主機現代化應URL用程式時，如何解決錯誤。

- 引擎： AWS 藍光時代和微焦點
- 組件：應用

如果您無法存取您建立並部署到 AWS 大型主機現代化執行環境的執行中 AWS 大型主機現代化應用程式，則可能需要在與執行階段環境相關聯的安全性群組上設定輸入規則。URL

常見原因

當您建立執行階段環境時，您提供的安全性群組 (包括預設安全性群組) 必須設定輸入規則，以允許從外部傳輸至已部署應用程式的VPC流量 (如果您要允許此類型的存取)。

解析度

檢查與執行階段環境關聯的 Amazon VPC 安全群組是否允許流量傳輸到適當應用程式連接埠上的環境。若要檢查安全性群組規則，請完成下列步驟：

1. 在開啟大 AWS 型主機現代化主控台。https://console.aws.amazon.com/m2/
2. 在左側導覽中，選擇 [環境]。
3. 選擇裝載您要連線之應用程式的執行階段環境。
4. 選擇「組態」。
5. 在 [安全性與網路] 中，選擇安全性群組。此連結會在 Amazon VPC 主控台中開啟安全群組的詳細資料。
6. 如有必要，請選擇 [編輯輸入規則]，並新增下列規則 (如果尚未存在)：

   Type

   自訂 TCP

連線埠

8196 或符合應用程式定義中指定之監聽器屬性的連接埠。如需詳細資訊，請參閱步驟 2：建立應用程式定義。

來源

您呼叫應用程式的來源 IP 位址。您可以從下拉菜單中選擇 MyIP。如果仍有逾時問題，請嘗試選擇「任何地方」IPV4 或「任何地方 IPV6」 在安全性群組上新增輸入規則之後，請務必停止應用程式並重新啟動應用程式。

如需詳細資訊，請參閱 Amazon VPC 使用者指南中的使用安全群組規則。

疑難排解：無法從主控台開啟 AWS 藍光深入解析

本頁說明如何解決未從 AWS 大型主機現代化主控台開啟的 Blu Insights 頁面的問題。

- 引擎： AWS 藍色時代
- 元件：藍光洞察

當您嘗試從 AWS 大型主機現代化主控台存取 Blu Insights 時，它不會開啟，而且新索引標籤會立即關閉。

常見原因

您用來存取藍光深入解析的角色沒有足夠的權限。

解析度

將IAM原則附加至角色，以允許其存取藍光深入解析。確保策略至少包含以下權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "m2:GetSignedBluinsightsUrl"
            ],
```

```
            "Resource": "*"
            }
    ]
}
```

確保更換region和account正確的 AWS 區域 和 AWS 帳戶。

疑難排解錯誤：環境狀況不良

本頁說明當您收到其中一個 AWS 大型主機現代化環境狀況不佳的通知時，如何解決錯誤。

- 引擎： AWS 藍光時代和微焦點
- 組件：環境

如果您收到通知，指出其中一個大 AWS 型主機現代化環境狀況不佳，這適用於您。您會透過下列其中一個來源收到通知：

- 不健康的環境狀態會顯示在您的大型主機現代化 AWS 主控台中。
- 關於不健康環境狀態的電子郵件通知。 AWS Health
- 您可以在 AWS Health 儀表板的 [您的帳戶健全狀況] 底下看到 AWS 大型主機現代化的相關事件。

常見原因

當您 AWS 帳戶中與 AWS 大型主機現代化環境相關聯的資源無法存取時，就會發生錯誤。造成此問題的常見原因是與環境相關的資源正在修改或刪除。

解析度

如需特定指引，請使用 AWSHealth 全狀況電子郵件中提供的錯誤碼，或透過您的大型主機現代化AWS主控台提供的錯誤碼。

錯誤代碼：

- 無法存取儲存

此錯誤表示環境的附加儲存 (Amazon Elastic File System FSx 案系統或 Amazon 檔案系統) 無法正確掛載。若要檢查健康狀況不良環境的詳細資訊，請完成以下步驟：

1. 在開啟大 AWS 型主機現代化主控台。https://console.aws.amazon.com/m2/
2. 選取狀況不良的環境，然後選擇組態。
3. 選擇附加的儲存體以檢視與此環境相關聯的儲存體資源。
4. 檢查與網路相關的組態，例如與儲存關VPC聯的安全群組、子網路和 Amazon。如果這些配置不正確，請嘗試恢復它們以解決此問題。

> **Note**
>
> 如果儲存區已刪除，則無法復原環境。在這種情況下，您應該考慮刪除不健康的環境。

疑難排解微焦點的授權問題

本頁說明如何使用 Micro Focus 執行階段引擎解決授權問題

- 引擎：微聚焦
- 組件：Amazon EC2

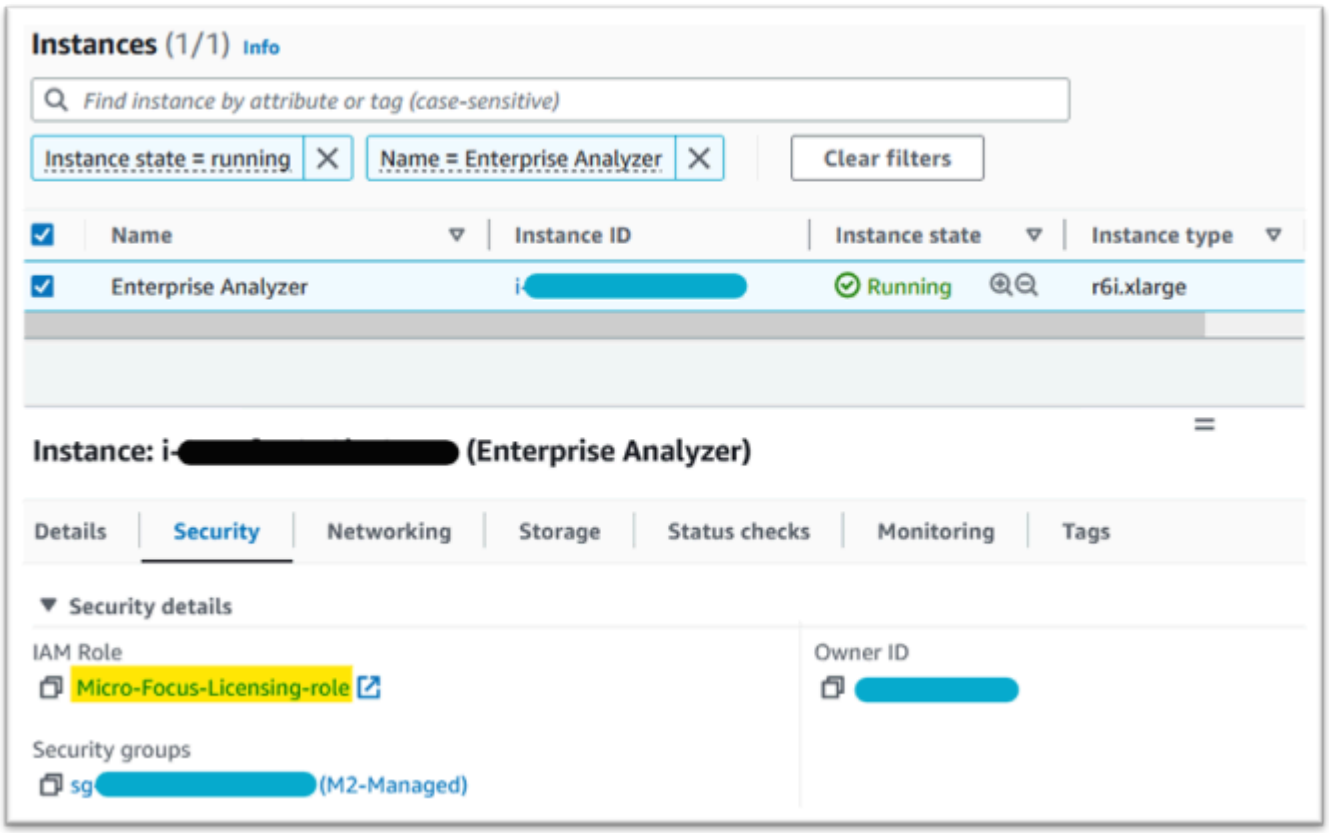
如果您在存取或使用時遇到問題AMIs，下列資訊可能會對您有所幫助。

主題

- 確認 Amazon EC2 執行個體具有IAM授權角色
- 使用可達性分析儀
- 執行授權守護程式
- 作業系統修補之後，Linux 上的企業伺服器或企業組建工具的授權問題

確認 Amazon EC2 執行個體具有IAM授權角色

這可以在 Amazon EC2 實例詳細信息的安全選項卡上進行檢查。您可以使用 「動作」下拉式功能表的「安全性選項」來變更此選項。



使用可達性分析儀

在「 AWS Network Manager 控制台」頁面上找到「Reachability Analyzer」。

在從 Amazon S3 端點建立的 Amazon EC2 執行個體和 Amazon S3 VPC 端點之間建立AMI和分析路徑。

如果 Amazon EC2 執行個體沒有網際網路存取權，請對所有 4 個端點重複路徑分析。

如需有關 Reachability Analyzer 的詳細資訊，請參閱「可 Reachability Analyzer」指南中的「可 Reachability Analyzer 入門」。

執行授權守護程式

在 Windows 企業開發人員從命令提示符使用以下命令：

```
“C:\Program Files (x86)\Micro Focus\Enterprise Developer\AdoptOpenJDK\bin\java” -jar
 "C:\Program Files (x86)\Micro Focus\Licensing\aws-license-daemon.jar"
```

並檢查輸出。忽略SLF4J訊息並尋找第一個例外狀況。

在企業分析器上，從命令提示符使用以下命令：

```
"C:\Program Files (x86)\Micro Focus\AdoptOpenJDK\bin\java" -jar "C:\Program Files
 (x86)\Micro Focus\Licensing\aws-license-daemon.jar"
```

並檢查輸出。忽略SLF4J訊息並尋找第一個例外狀況。

在 Linux 上運行：

```
java -jar /var/microfocuslicensing/bin/aws-license-daemon.jar
```

忽略SLF4J訊息並尋找第一個例外狀況。

例如，如果 Amazon S3 資源無法使用，則例外情況如下：

```
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.

Exception in thread "main" software.amazon.awssdk.services.s3.model.S3Exception: Access
 Denied (Service: S3, Status Code: 403, Request ID: P6
```

例外訊息會指出哪些資源不可用。將組態值與本主題中顯示的值進行比較。

作業系統修補之後，Linux 上的企業伺服器或企業組建工具的授權問題

如果您在作業系統修補之後遇到 Linux 上的企業伺服器或企業組建工具的授權問題，請下載並執行修補程式指令碼來更新授權協助程式。為此，請在命令提示符下使用以下命令：

```
sudo curl https://d148y999krizvm.cloudfront.net/patch/v8/linux/patch.sh -o /var/
microfocuslicensing/bin/patch.sh
sudo chmod +x /var/microfocuslicensing/bin/patch.sh
sudo /var/microfocuslicensing/bin/patch.sh
sudo ./startmfcesd.sh
```

> **Note**
>
> 即使下載路徑適用於版本 8，此修補程式指令碼也適用於版本 9。

AWS 大型主機現代化使用者指南的文件歷史記錄

下表說明 AWS 大型主機現代化的文件版本。

變更	描述	日期
彙編程序轉換 mLogica	AWS 大型主機現代化程式碼轉換 mLogica 是一項大 AWS 型主機現代化功能，可自動將 z/OS 大型主機彙編程式碼轉換為. COBOL	2024年7月22日
應用程序測試 GA 發布	應用程式測試的一般可用性文件。 AWS 大型主機現代化應用程式測試可為您的移轉專案提供自動化功能等效測試。此版本包含資料保護頁面、主控台工作流程，以及自預覽以來對其他文件頁面的更新。	2024年6月12日
更新微焦點管理運行時教程	本教學課程 AWS 說明如何使用 Micro Focus 執行階段引擎，在大型主機現代化管理的執行階段環境中部署及執行 CardDemo 範例應用程式。	2024年2月5 日
AWS 藍光時代運行時和現代化工具版本 3.9.0 的發行說明。	此版本的 AWS Blu Age Runtime 和現代化工具專注於跨產品的多項橫向增強功能，致力於提高高可用性架構的性能，以及將作業執行提升到一個新的水平的新功能。	2023 年 12 月 18 日
在大型主機和 AWS	發佈新功能，可將檔案從來源大型主機傳輸到. AWS	2023 年 11 月 27 日

管理應用程式的交易	發佈新功能，用於顯示和編輯 AWS 大型主機現代化應用程式的交易。	2023 年 10 月 16 日
AWS 藍光時代運行時和現代化工具版本 3.6.0 的發行說明。	此版本的 AWS Blu Age Runtime 和現代化工具為 ZoS 和 AS4 00 舊版遷移提供了新功JCL能，主要面向擴展CICS支持機制，補充功能，優化並發和大批量功能的性能以及添加功能。 multi-data-source	2023 年 8 月 4 日
您現在可以在應用程式停止時部署新版本的應用程式。	先前，若要部署新版本的應用程式，您必須刪除已部署的版本。現在，您可以停止部署的版本並部署新版本。	2023 年 7 月 26 日
AWS 藍光時代運行時打包，方便 Amazon EC2 部署	AWS 使用 AWS Blu Age 執行階段的大型主機現代化現在可提供更大的彈性，可在您的 Amazon EC2 執行個體上設定完整堆疊和部署。 AWS 帳戶	2023 年 7 月 6 日
單一登入 AWS 藍光時代藍光洞察。	AWS 藍光時代藍光見解可從 AWS Management Console 通過單一登錄獲得。	2023 年 3 月 31 日
GA 版本	GA 發行的大 AWS 型主機現代化使用者指南。	2022 年 6 月 8 日
初始版本	AWS 大型主機現代化使用者指南的初始版本 (公開預覽)。	2021 年 11 月 30 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。